VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Marek Navrátil

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MONITOROVACÍ SYSTÉM KOTELNY
MONITORING SYSTEM OF BOILER ROOM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Marek Navrátil**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Zdeněk Bradáč, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Marek Navrátil ***ID:*** 144908
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Monitorovací systém kotelny

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a vytvořte elektronický systém pro monitorování kotelny na tuhá paliva. Navrhněte systém založený na vhodné mikrokontrolérové platformě, který umožní monitorovat stav vybavení kotelny včetně obsahu nebezpečných plynů. Navrhněte a zrealizujte elektronický systém, který vybavte nezbytnými rozhraními, sensory a akčními členy. Vytvořte nezbytné programové vybavení včetně vizualizace. Otestujte a demonstrujte plnou funkčnost.

1. Zpracujte literární rešerši.
2. Navrhněte a popište vlastní koncepci systému.
3. Realizujte vlastní obvodové řešení, realizujte DPS, oživte.
4. Vytvořte funkční programové vybavení.
5. Ověřte a demonstrujte funkčnost systému a vyhodnoťte výsledky.

DOPORUČENÁ LITERATURA:

1. ZEZULKA, F. Prostředky průmyslové automatizace. VUTIUM. VUTIUM. Brno: VUTIUM, 2004. 176 s. ISBN: 80-214-2610-1.

2. Dle doporučení vedoucího práce.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 14.5.2018

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Tato diplomová práce se zabývá návrhem monitorovacího systému pro kotelny na tuhá paliva. V první kapitole jsou popsány senzory, které se nejčastěji v kotelnách používají, včetně popsání principu činnosti. Hlavním obsahem práce je návrh vlastního systému pro monitorování kotelen. Nejprve je zvolena vhodná struktura celého systému, poté jsou podrobně popsány návrhy jednotlivých částí zařízení elektronické zapojení a výběr vhodných komponentů. Poslední kapitola popisuje vytvořené programové vybavení zařízení. Výsledkem práce jsou podklady pro výrobu navrženého systému a vytvořený obslužný program.

KLÍČOVÁ SLOVA

Monitorovací systém kotelny, kotelna, Raspberry PI 3, Arduino Nano, senzory plynů CO a CO_2, snímače vlhkosti, tlaku a teploty, python, flask, MySql

ABSTRACT

This diploma thesis deals with the design of a monitoring system for solid fuel boilers. The first describes the sensors most commonly used in boilers room including a description of the principle of operation. The main content of the thesis is the design of the own boiler monitoring system. First the appropriate structure of the entire system is selected followed by the detailed design of individual parts of the device electronic wiring and selection of suitable sensors. The result of the thesis is the basis for the production of the proposed device and created service program.

KEYWORDS

Monitoring system of boiler room, boiler room, Raspberry PI 3, Arduino Nano, gas sensors CO and CO_2, sensors of humidity, pressure and temperature, python, flask, MySql

NAVRÁTIL, Marek. *Monitorovací systém kotelny*. Brno, 2018, 75 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Monitorovací systém kotelny“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu Diplomové práce panu doc. Ing. Zdeňkovi Bradáčovi, Ph.D. za podnětné návrhy k práci.
Dále bych poděkoval paní Mgr. Michaele Juráňové za gramatickou úpravu textu a slečně Lucii Hradilové za trpělivost.

Brno ....

podpis autora(-ky)

OBSAH

SEZNAM OBRÁZKŮ

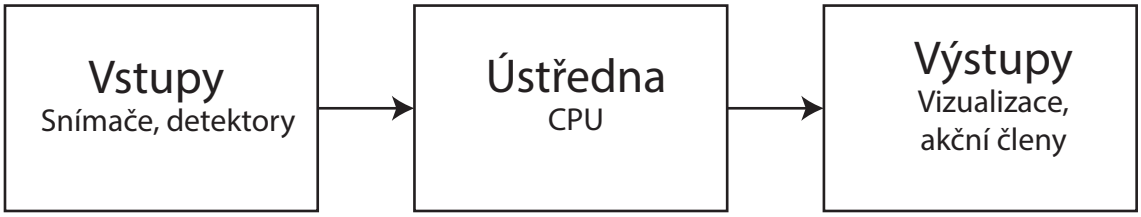
SEZNAM TABULEK

ÚVOD

Kotelna je důležitá místnost v našich domácnostech, neboť zde vzniká teplo, které prohřívá domy a lidská těla v zimních měsících. Jak je po staletí známo, oheň je dobrý sluha, ale zlý pán, proto je velice důležité mít jej pod kontrolou. S rozvojem elektroniky a počítačových sítí je již možné tento proces monitorovat na dálku. K tomu slouží „Monitorovací systémy kotelen“.

1 MONITOROVACÍ SYSTÉMY

Jsou informační systémy zabývající se sběrem, tříděním a ukládáním informací o sledovaných jevech a následným vyhodnocováním. Jádro každého monitorovacího zařízení je tvořeno ústřednou, která sbírá data ze vstupních snímačů a detektorů. Jejich činností je sledování fyzikálních jevů (teplota, tlak). Data jsou v ústředně zaznamenávána a upravována pro další zpracování jako je vizualizace průběhů a ovládání výstupů. Základní blokové schéma monitorovacího systému je zobrazeno na obr. 1.1 [1].



Obr. 1.1: Obecné schéma monitorovacího systému

1.1 Snímače

V monitorovacích systémech kotelen na tuhá paliva bývají nejčastěji zaznamenávány průběhy těchto veličin [2] [3]:

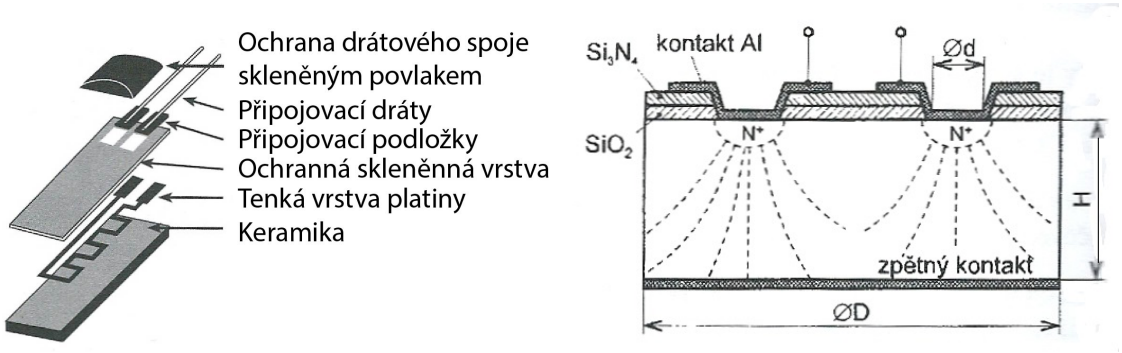
- Teploty kotle
- Teploty užitkové vody
- Tlak otopného média
- Koncentrace oxidu uhelnatého
- Koncentrace oxidu uhličitého
- Vlhkosti
- Detekce plamene

1.1.1 Snímače teploty

Pro měření teploty se využívá určitých fyzikálních veličin závislých na teplotě. Při měření se porovnává teplota zkoumaného tělesa se standardizovanou stupnicí. Senzory teploty se dle fyzikálního principu dělí na odporové, termoelektrické, polovodičové, chemické, optické, magnetické atd. Dále je možné tyto snímače dělit podle styku s měřeným objektem na dotykové a bezdotykové. Při měření teploty v kotelnách se budou hlavně využívat dotykové senzory.

Odporové senzory

Tyto typy senzorů jsou nejčastěji používanými prostředky pro měření teploty z důvodů dobré stability a přesnosti. Hlavní část je tvořena odporovým materiálem, který bývá vyráběn buď z čistého kovu (platina, nikl), nebo z polovodičového monokrystalu, obě varianty zobrazuje obr. 1.2. V závislostí na teplotě se mění odpor této struktury a tím dochází ke změně úbytku připojeného napětí [4].



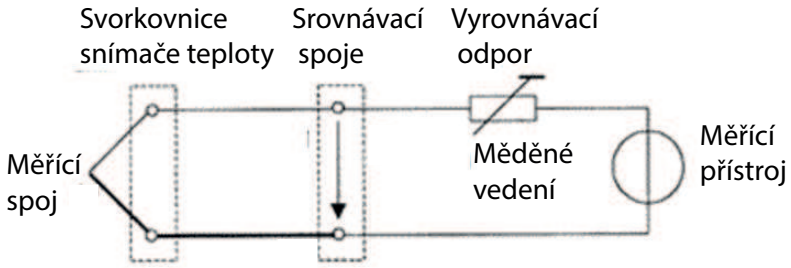
Obr. 1.2: Odporové čidlo platinové nalevo a monokrystalické napravo [4]

Odporová čidla se dříve vyráběla v drátovém provedení, dnes převládá vrstvová technologie. Na keramické destičce je nanesena vrstva odporového materiálu. Nanesení je prováděno různými metodami např. sítotiskem, napařováním nebo naprašováním. Nastavení základní hodnoty odporu se provede vypálením odporové cesty pomocí laseru.

Polovodičová odporová čidla bez PN přechodu pracují na principu kuželovitého rozptylu proudu mezi dvěma elektrodami a využívají závislost pohyblivosti volných nosičů náboje na odporu. Tento odpor se zvyšuje s rostoucí teplotou. Rozšířeným polovodičovými senzory jsou termistory, které se dělí na NTC (negastory) s negativním součinitelem odporu a na PTC (pozistory) s pozitivním součinitelem odporu [4].

Termoelektrické senzory

Pracuji na principu vzniku termoelektrického napětí ve styku dvou různých kovů nebo polovodičů, jejichž konce jsou umístěny v prostředí s rozdílnými teplotami (obr. 1.3). Příčinou vzniku termoelektrického napětí je difuze nosičů náboje z teplejší oblasti do chladnější a vznik vnitřního elektrického pole, které pak další difuzi brání. Vyhodnocovací obvody si musí poradit s rušivými signály jako je kolísání srovnávací teploty a vliv odporů vedení. Konstantní teploty srovnávacího spoje je docíleno buď regulujícím termostatem, nebo vložením kompenzační krabice s teplotně závislým můstkem. Tyto senzory jsou často používány pro svou jednoduchost, tepelnou odolnost a široký pracovní rozsah teplot. Vzhledem ke své malé hmotnosti a setrvačnosti jsou vhodné pro měření rychlých změn teploty [4].



Obr. 1.3: Konstrukce termočlánku [4]

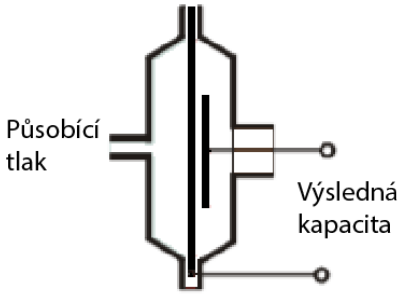
1.1.2 Snímače tlaku

U těchto senzorů se převážně využívá princip převodu síly, tlaku a tíhy na deformaci. Využívají tedy fyzikálních účinků síly.

Základem tlakových snímačů bývá tlakoměrný prvek (membrána, trubice, vlnovec), u kterého dochází vlivem působící síly k deformaci. Na tento prvek navazuje vhodný senzor s elektrickým výstupem, který vyhodnocuje deformaci způsobenou změnou tlaku. Ten je vyhodnocován změnou polohy nebo mechanickým napětím. Nejpoužívanějšími jsou senzory tenzometrické, piezoelektrické a kapacitní.

Kapacitní senzory tlaku

Kapacitní čidlo je principiálně jednoduché zobrazeno na obr. 1.4. Základem jsou dvě elektrody jedna je tvořena membránou a při působení tlaku u ní dochází ke změně polohy. Druhá je pevně uchycena. Změna vzdálenosti mezi elektrodami se projeví změnou kapacity [5].

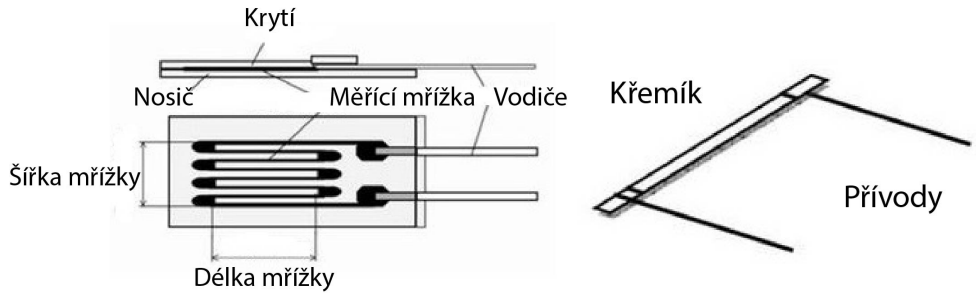


Obr. 1.4: Principiální schéma kapacitního čidla tlaku [5]

Odporové tenzometry

U kovových nebo polovodičových tenzometrů dochází ke změně ohmického odporu, vlivem mechanických deformací při působení tlaku. U kovových tenzometrů, obr. 1.5 vlevo, se mění průřez a délka drátku měřící mřížky, která sleduje deformaci měřeného povrchu. Mřížky jsou navinuty ze slabého konstantanového drátku, nebo leptány z folie. Odporové tenzometry jsou využívány v aplikacích, které vyžadují velkou přesnost [4].

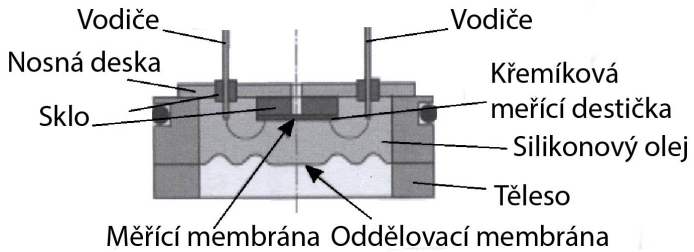
U polovodičových odporových tenzometrů dochází vlivem tlaku ke změně pohyblivosti nosičů náboje. Tyto senzory jsou vyráběny z křemíku s vhodnou krystalografickou strukturou. Zobrazeno na obr. 1.5 vpravo. Přednosti polovodičových tenzometrů jsou ve vyšší citlivosti a malých rozměrech. Nevýhodnou je nutná kompenzace tepelné závislosti pomocí měřicích můstků [4].



Obr. 1.5: Odporový tenzometr kovový nalevo a polovodičový napravo [4]

Piezoelektrické senzory tlaku

Pro svou činnost využívají piezoelektrického jevu. Kdy při působení mechanických deformací dochází u některých krystalů ke vzniku elektrického náboje. Piezoelektrické snímače se používají zejména k měření dynamických tlaků, výhodou jsou jejich malé rozměry, jednoduchost a široké frekvenční spektrum. Struktura zobrazena na obrázku 1.6 [4].



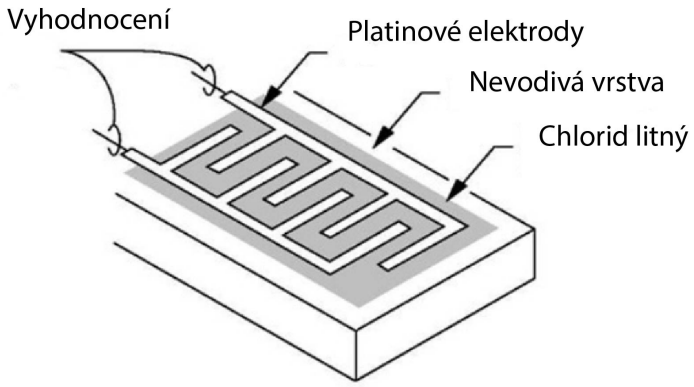
Obr. 1.6: Piezoelektrický snímač tlaku [4]

1.1.3 Snímače vlhkosti

Vlhkost udává, jaké množství vody v plynném stavu je obsaženo v daném množství vzduchu. Rozlišujeme absolutní a relativní vlhkost. Absolutní vlhkostí se vyjadřuje hmotnost vodní páry obsažené v jednotce objemu vzduchu. Relativní vlhkost udává poměr mezi okamžitým množstvím vodních par ve vzduchu a množstvím par, které by měl vzduch o stejném tlaku a teplotě při plném nasycení. Relativní vlhkost se udává v procentech. Existují různé druhy elektronických senzorů vlhkosti. Nejpoužívanějšími jsou odporové, kapacitní a infračervené.

Odporové senzory vlhkosti

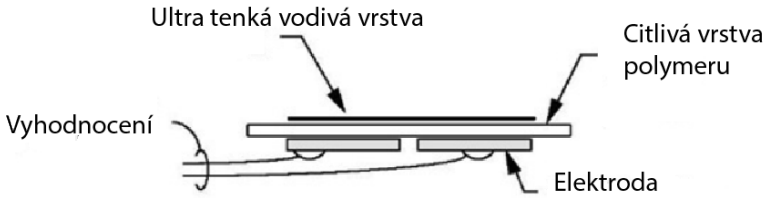
U tohoto typu senzoru dochází ke změně elektrické vodivosti v závislosti na vlhkosti. Základem je tenká izolační trubička potažená skelnou tkaninou a napuštěná chloridem litným, který dobře pohlcuje vodní páru ze vzduchu. Dále jsou strukturou vedeny dvě oddělené platinové elektrody zobrazeno na obr. 1.7. Protékající proud elektrodami způsobuje zahřátí chloridu lithného a z jeho povrchu dochází k odpařování pohlcené vody. Nastává zvýšení vodivosti a teploty struktury. Teplota je přímo úměrná relativní vlhkosti. Odporové senzory vlhkosti disponují vysokou přesností a stabilitou [6].



Obr. 1.7: Odporový snímač vlhkosti [6]

Kapacitní senzory vlhkosti

Využívají principu absorpce vodních par v polymerních materiálech. Senzor je tvořen dvěma vodivými deskami, které jsou odděleny polymerním materiálem citlivým na vlhkost. Materiál absorbuje vodu, a tím dochází ke změně kapacity. Aby se zvýšila schopnost materiálu absorbovat a uvolňovat vzdušnou vlhkost, bývají obě vodivé desky umístěny na spodní straně materiálu reagujícího na vlhkost. Na horní stranu se umístí ultra tenká deska vodivého materiálu, která dovoluje lepší průnik vlhkosti. Principiální schéma zobrazeno na obr. 1.8. Kapacitní senzory vlhkosti jsou teplotně nezávisle s krátkou dobou odezvy [6].



Obr. 1.8: Kapacitní snímač vlhkosti [6]

1.1.4 Snímače oxidu uhelnatého

Oxid uhelnatý (CO) je bezbarvý, hořlavý plyn bez chuti a zápachu. Vzniká nedokonalým spalováním uhlíkatých materiálů viz rovnice 1.1:

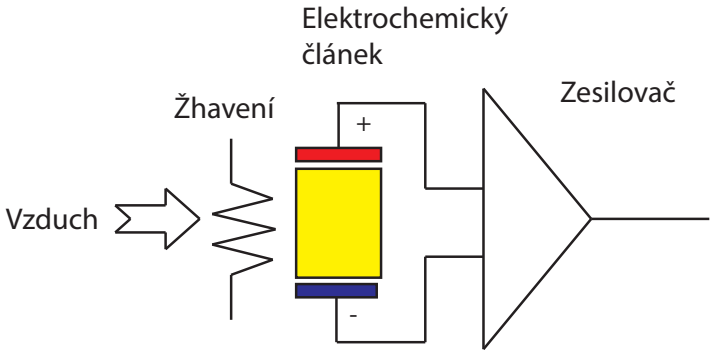
$$2\mathrm{C} + \mathrm{O}_2 \rightarrow \mathrm{CO}. \tag{1.1}$$

Při vdechnutí se CO váže na červené barvivo (hemoglobin) a zabraňuje tak krvi vázat a rozvádět kyslík po těle. Při vysoké koncentraci může být pro člověka smrtelný, menší koncentrace ublíží především lidem trpícím kardiovaskulárním onemocněním. Podrobnější přehled účinků plynu na člověk viz tab 1.1 a [7].

Tab. 1.1: Vliv koncentrace CO na zdraví člověka [7]

Koncentrace CO [ppm]	***Účinky na člověka***
200	Nevolnost po 2-3 hodinách.
400	Životu nebezpečný stav po 3 hodinách.
800	Bezvědomí, smrt během 2-3 hodin.
1600	Smrt během 1 hodiny.
6400	Smrt během 10-15 minut.

Snímače oxidu uhelnatého pracují na principu elektrochemického článku. Tento článek je vyhříván pomocným žhavením na pracovní teplotu. Mezi elektrodami dochází k chemickým reakcím a na elektrodách vzniká elektromotorická síla. Měřením této síly pomocí speciální elektroniky se pak zjišťuje koncentrace CO zobrazeno na obr. 1.9 [8].



Obr. 1.9: Princip činnosti snímače CO [8]

1.1.5 Snímače oxidu uhličitého

Oxid uhličitý (CO_2) je bezbarvý plyn bez chuti a zápachu. Jeho molekula je tvořena jedním atomem uhlíku a dvěma atomy kyslíku. Vzniká již klasickým spalováním uvedeno v rovnici 1.2:

$$C + O_2 \rightarrow CO_2. \tag{1.2}$$

Je hlavním plynem přispívajícím k narůstání množství skleníkových plynů v atmosféře. Běžná koncentrace oxidu uhličitého je v atmosféře nízká a nepředstavuje pro člověka přímé riziko. Nebezpečí vzniká při vyšších koncentracích tohoto plynu. Krátkodobá expozice vyššího množství oxidu uhličitého může způsobovat bolesti hlavy, závratě, dýchací potíže a jiné. Vysoká koncentrace pak může přivodit křeče, kóma a smrt podrobněji zobrazeno v tab. 1.2 a [7].

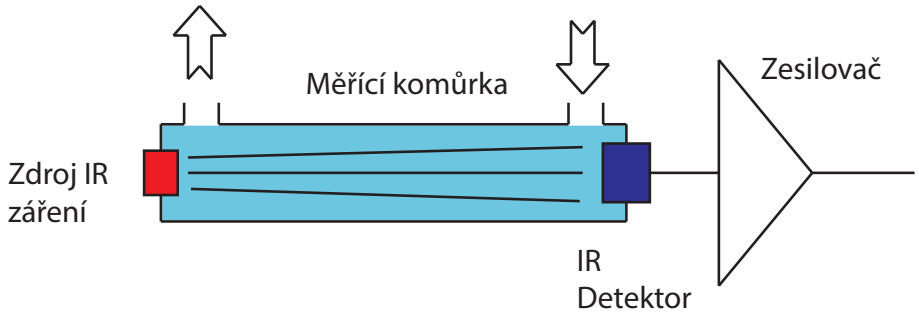
Tab. 1.2: Vliv koncentrace CO_2 na zdraví člověka [7]

Koncentrace CO_2 [ppm]	***Účinky na člověka***
360-400	Čerstvý vzduch v přírodě
800-1000	Doporučená úroveň CO_2 ve vnitřních prostorách
> 1000	Nastávají příznaky únavy a snižování koncentrace
5000	Maximální bezpečná koncentrace bez zdravotních rizik

Senzory pro měření koncentrace oxidu uhličitého využívají několik principů. Nejrozšířenější snímače pracují na základě absorpce infračerveného záření (NDIR). Dále se pro detekci využívá elektroakustického a elektrochemického principu, který je podobný jako u měření koncentrace CO viz kapitola 1.1.4.

Snímače NDIR

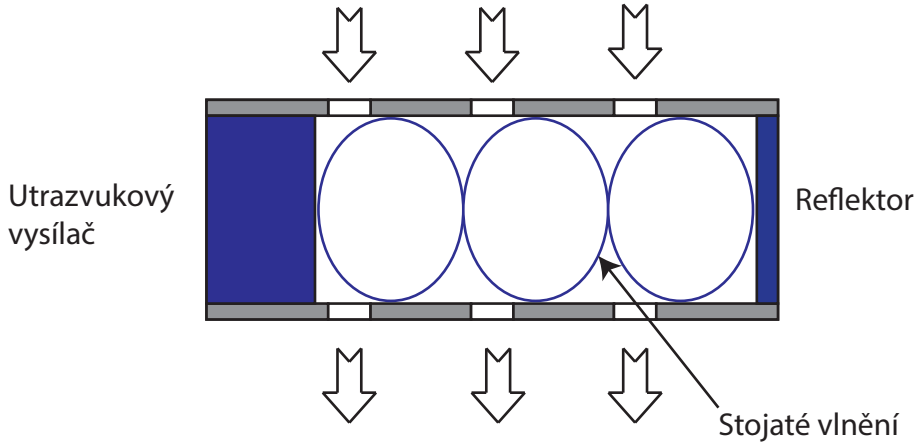
U těchto snímačů se využívá útlumu infračerveného záření v plynu. Snímač se skládá ze zdroje infračerveného záření, světlo-vodné trubice a infračerveného detektoru s filtrem. Signál z detektoru je zesilován a pak se pomocí elektroniky vyhodnocuje útlum, který odpovídá koncentraci CO_2 ve vzduchu, jak je zobrazeno na obr. 1.10 podrobnější informace [9].



Obr. 1.10: NDIR snímač CO_2 [9]

Elektroakustická čidla

Tyto snímače vyhodnocují změny kmitočtu ultrazvuku v mechanickém rezonátoru. Elektronika vyhodnocuje změny kmitočtu ultrazvukových vln v závislosti na koncentraci CO_2 ve vzduchu a určuje aktuální hodnotu měření obr. 1.11 [8].



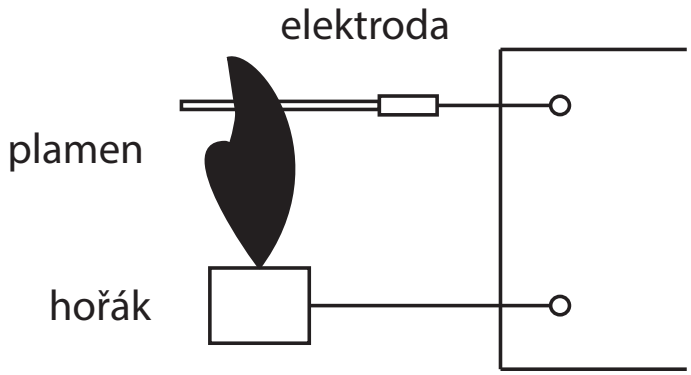
Obr. 1.11: Elektroakustický snímač CO_2 [8]

1.1.6 Detektory plamene

Pro detekci plamene se používají dva základní typy detektorů, detektory ionizační a optické.

Ionizační detektory

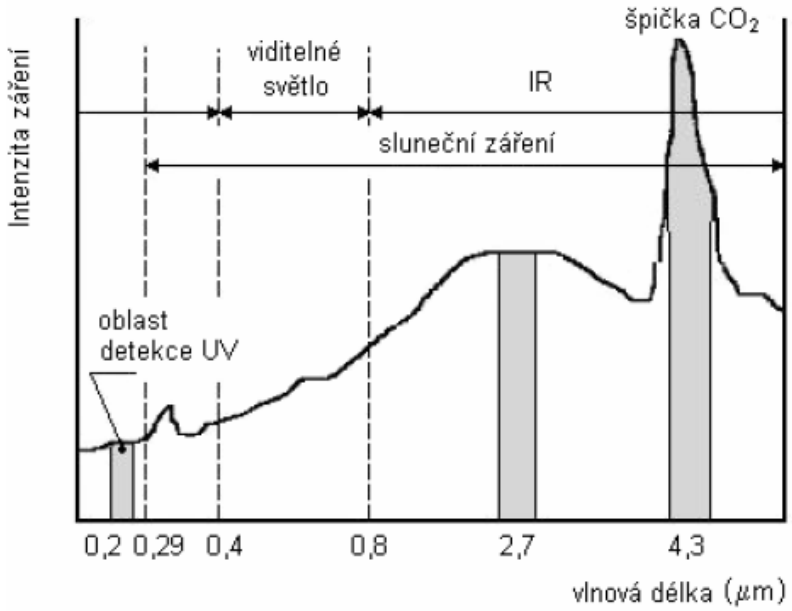
Pracují na principu snímání elektrického proudu mezi dvěma elektrodami. Jedna elektroda ve tvaru malé tyčinky je umístěna v plamenu a druhá je spojená s hořákem obr. 1.12. U elektrody nad hořákem dochází k ionizaci neutrálních částic, které zvyšují velikost procházejícího proudu. Podrobnější informace viz [10].



Obr. 1.12: Ionizační snímač plamene [10]

Optické detektory

Při použití optických detektorů se vyhodnocuje elektromagnetické záření plamene v různých oblastech spektra. Jedná se o ultrafialovou, viditelnou či infračervenou část. Na obr. 1.13 je znázorněno spektrum vyzařované pří hoření. Zvýrazněné části zachycují oblasti spektra, ve kterých obvykle pracují tyto senzory[11].



Obr. 1.13: Spektrum plamene [11]

1.2 Zpracování dat

Data získaná ze senzoru je potřeba zpracovat do takové formy, aby se daly použít pro výstupní vizualizaci či ovládání připojených výstupních zařízení. Obvyklé fyzické prostředky, které k tomu účelu slouží, můžeme nazvat řídícími systémy. Jsou to univerzální prostředky, dnes realizovatelné na elektronickém principu. Mezi ně patří programovatelná logická pole, mikroprocesory, mikrořadiče a další [12].

Jedná-li se o málo výpočetně náročná operace, používají se v těchto aplikacích mikrokontrolery. Jejích velkou výhodnou jsou malé rozměry a jednoduchost. Po připojení napájení, vstupů a výstupů dostáváme plnohodnotnou výpočetní jednotku, jejímž příkladem mohou být 8, 16 a 32 bitové mikrokontrolery od výrobců Atmel, Motorola atd.

Při zpracovávání dat, které slouží pro řízení technologických procesů, je možné použití programovatelných logických automatů. Jejich výhoda spočívá v tom, že jejich periférie jsou uzpůsobeny pro napojení na průmyslové procesy.

1.3 Zobrazení dat

Monitorovací systém pro kotelny může spadat pod elektronickou požární signalizaci (EPS). Výstupní data bývají nejčastěji poskytována pro dohledová centra, kde jsou vyhodnocována člověkem, který v případě zjištění závady reaguje zásahem. Dále systém může být navrhnut tak, že samostatně reaguje na nebezpečné hodnoty a může automaticky ovládat přípojená ochranná, signalizační či hasící zařízení viz [1].

Využívají se také jako součást automatického řízení chodu budovy či HVAC. Senzory mohou být připojené k PLC a sloužit rovnou k automatické regulaci a monitorování pomocí HMI/SCADA [12].

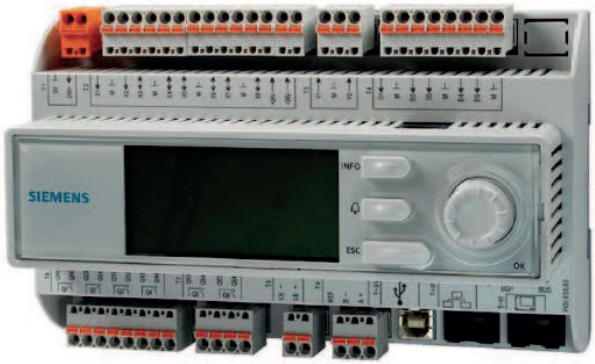
Pro každodenní využití, například v domácnostech, je výhodné zaznamenané hodnoty distribuovat na webový server, kde je možná vzdálená a jednoduchá správa přes chytrý telefon.

1.4 Komerčně dostupné systémy

Jedním z komerčně dostupných systémů je Poruchová signalizace Kotelník v1.0 zobrazena na obr. 1.14 od společnosti Siemens. Podrobnější informace viz manual [3]. Zařízení je určené pro signalizaci chyb u zdrojů tepla a monitoruje následující veličiny:

- Tlak v systému
- Teplotu v systému a prostoru kotelny
- Hladinu vody zdrojů popř. zaplavení prostoru

- Signalizace úniku plynu, chladiva, CO
- Počet výpadků napájení



Obr. 1.14: Komerční systém Kotelník v1.0 [3]

Zařízení obsahuje 4x analogový vstup, 9x digitální vstup, napájení 24 V a komunikaci pomocí Ethernetu. Dále obsahuje digitální rozpínací a spínací kontakty. K zařízení je možné připojit GSM modul pro zasílání SMS zpráv.

Další výrobce zabývající se problematikou monitorování kotelen je Remforce. Firma nabízí datový kolektor REM1 zobrazeno na obr. 1.15. Zařízení disponuje 4x vstupy pro měření teploty. Dále je možné připojit 8x digitální vstup a 4x analogové vstupy. Naměřená data jsou ukládána do cloudu. Tyto data je poté možné sledovat v reálném čase pomocí počítače či chytrého telefonu [13].

Obr. 1.15: Dataloger od firmy Remforce [13]

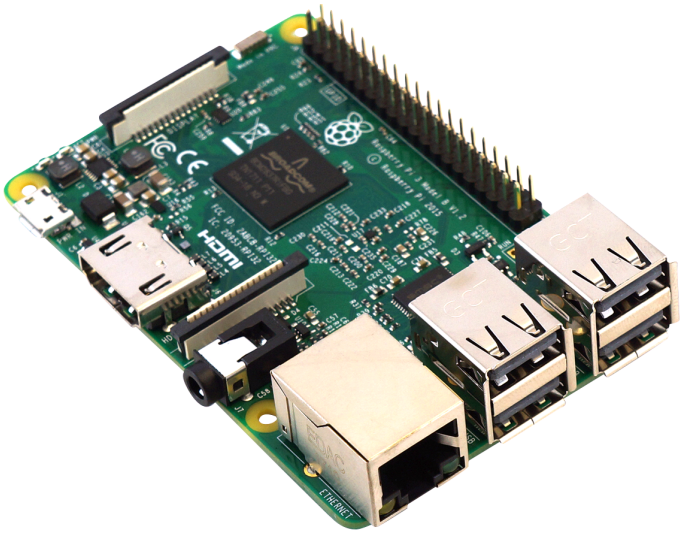
2 REALIZACE SYSTÉMU

2.1 Požadavky na systém

Vytvořený monitorovací systém kotelny by měl být schopen měřit teplotu z několika různých míst. Důležitá je teplota otopného média, která upozorňuje na překročení teploty nebo na nebezpečí zamrznutí. Dále teploty prostoru, která upozorňuje na překročení maxima a umožňuje např. spuštění ventilátoru. Hlavní monitorovací veličinou v topných systémech je koncentrace nebezpečných plynů, která musí být v systému zahrnuta. V případě kotelen na tuhá paliva jsou to plyny CO a CO_2. Naměřená data by měla být zaznamenávána na webový server. Zařízení by mělo obsluhovat výstupní zařízení, které bude reagovat na nebezpečné stavy v prostoru kotelny, a tak bude možné předejít újmě na zdraví a majetku.

2.2 Základní koncepce

Dle požadavků zmíněných v kapitole 2.1 byl zvolen vhodný CPU, jednodeskový počítač Raspberry PI 3, dále v textu uváděno jako Raspberry. Jedná se o počítač velikosti kreditní karty (obr. 2.1) s ovladatelnými vstupně/výstupními porty GPIO. Použitý mikroprocesor je z rodiny ARM, takže jeho výkon je srovnatelný s výkonem chytrého telefonu. Na Raspberry je možné provozovat různé distribuce Linuxu a Windows 10 IoT. Používá se jako multimediální přehrávač videa, hudby nebo jen pro přístup k internetu. Hlavní výhodou modelu PI 3 je zabudovaný Wifi a Bluetooth modul. Podrobnější informace viz [14].

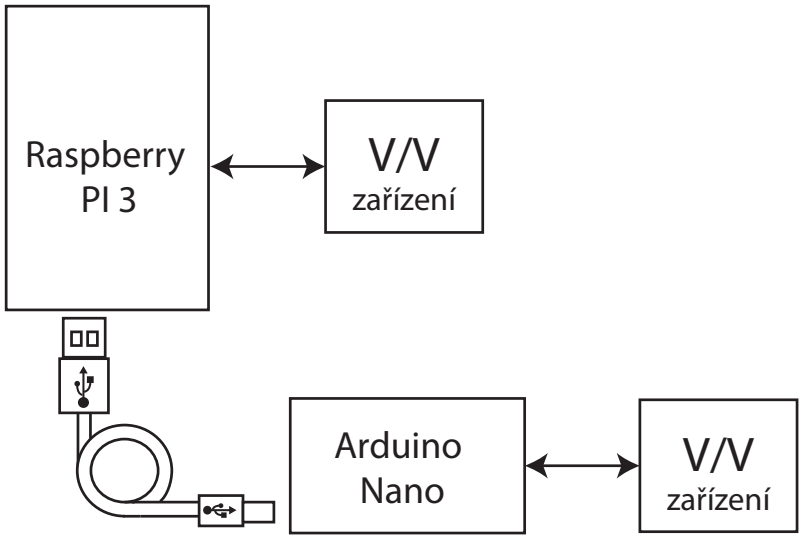


Obr. 2.1: Jednodeskový počítač Raspberry PI 3 [14]

Raspberry bude zpracovávat informace ze senzorů. Poběží na něm webový server pro vizualizaci dat. Při práci se vstupně/výstupními porty GPIO je nutno počítat s omezením. Maximální velikost procházejícího proudu na jeden pin GPIO je 16 mA, odebíraný proud přes všechny GPIO porty je 50 mA a velikost připojeného napětí činí 3,3 V. Z 5 V výstupu je možné odebírat proud až 1 A. Dále u GPIO portů chybí analogový vstup, lze pracovat jen s digitálním vstupem. Výhoda u GPIO portů je pak v sériové sběrnici, I^2C a sběrnici 1-Wire, která je řešená pomocí softwaru. Absence analogových vstupů a výstupů s větším proudovým odběrem lze kompenzovat např. připojením I^2C expandérů či jiných elektronických obvodů napájených externím zdrojem.

Pokud bychom ponechali obsluhu jednotlivých senzorů pouze na Raspberry, setkali bychom se s problémem obsluhy snímačů v reálném čase. V multiprocesorovém systému je nevhodné, aby sám procesor prováděl synchronní pulling, protože by takovýto program spotřebovával část procesorového času úplně zbytečně. Navíc by nebylo zaručeno přidělení procesoru danému programu, proto je mnohem výhodnější přenechat obsluhu senzorů mikrokontroleru, který zaručí obsluhu v reálném čase. K Raspberry jsou potom už jen zasílány naměřené hodnoty.

Vzhledem k výše popsaným skutečnostem bylo rozhodnuto, že obsluhu vybraných připojených senzorů bude zajišťovat mikrokontroler Arduino Nano, dále v textu uváděno jako Arduino. Tak je získán snadno rozšiřitelný modulární systém. Koncepce celého systému je zobrazena na obr. 2.2.



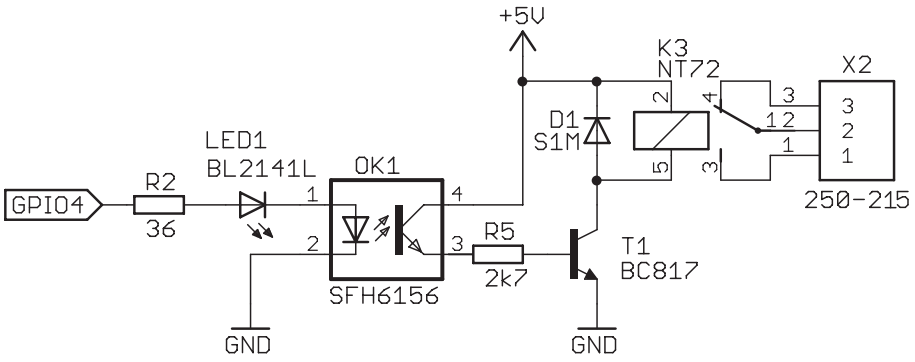
Obr. 2.2: Navržená koncepce Raspberry PI 3 a Arduino Nano

K Raspberry jsou připojena teplotní čidla a ovládací relé. Arduino obsluhuje senzory plynů a vlhkosti.

2.3 Raspberry periferie

2.3.1 Relé výstupy

Pro spínání větších proudů bylo vybráno přepínací relé NT72. Jedná se o 5 V relé, které je schopno spínat zařízení s proudovým odběrem až do 10 A, což je dostatečné pro ovládání ventilátorů, vzduchových odsavačů či signalizačních zařízení, které se budou v kotelnách využívat. Pro připojené relé byl sestaven elektronický obvod, který je zobrazen na obr. 2.3.



Obr. 2.3: Připojení relé k Raspberry

Relé je napájeno z 5 V výstupního pinu GPIO Raspberry, který umožňuje velikost odebíraného proudu až 1 A. Celkový proud pro sepnutí relé vychází z rovnice:

$$I_{\mathrm{RELE}} = \frac{U_{\mathrm{RASP5V}}}{R_{\mathrm{VIN}}} = \frac{5}{56} = 89 \text{ mA}, \tag{2.1}$$

kde U_{RASP5V} je výstupní napětí pinu 5 V a R_{VIN} je hodnota odporu vinutí vybraného relé dle katalogového listu.

Relé je spínáno pomocí tranzistoru T1 a jako další ochrana je mezi tranzistor a výstupní pin přidán optočlen a signalizační dioda. Hodnota předřadného odporu pro LED je dána vztahem:

$$R_{\mathrm{R2}} = \frac{U_{\mathrm{GPIO}} - U_{\mathrm{LED}} - U_{\mathrm{OPTO}}}{I_{\mathrm{LED}}} = \frac{3,3 - 1,8 - 1,25}{7 \cdot 10^{-3}} = 35 \ \Omega, \tag{2.2}$$

kde U_{GPIO} je výstupní napětí GPIO, U_{LED} je úbytek napětí na diodě, U_{OPTO} je úbytek napětí na optočlenu, I_{LED} je hodnota procházejícího proudu určená maximálním proudem LED.

Výstupní proud optočlenu je potom roven:

$$I_{\mathrm{C}} = I_{\mathrm{LED}} \cdot CTR = 7 \cdot 10^{-3} \cdot 1,5 \cong 11 \text{ mA}, \tag{2.3}$$

kde I_{LED} je proud LED, CTR je proudový přenosový poměr pro optočlen pro hodnotu I_{LED}.

Odpor přechodu emitor/kolektor u tranzistoru v optočlenu je potom dán dle Ohmova zákona:

$$R_{\mathrm{OPTO}} = \frac{U_{\mathrm{RASP5V}} - U_{\mathrm{SAT}}}{I_{\mathrm{C}}} = \frac{5 - 0,4}{7 \cdot 10^{-3}} = 45 \ \Omega, \tag{2.4}$$

kde U_{RASP5V} je napájecí napětí, U_{SAT} je saturační napětí tranzistoru.

Pro spínání zátěže o 89 mA je potřeba nejmenší velikost bázového proudu pro tranzistor dle vztahu:

$$I_{\mathrm{B}} = \frac{I_{\mathrm{C}}}{h_{\mathrm{FE}}} = \frac{89}{160} = 0,55 \text{ mA}, \tag{2.5}$$

kde h_{FE} je proudový zesilovací činitel tranzistoru.

Hodnota bázového odporu pro tranzistor T1:

$$R_{\mathrm{R5}} = \frac{U_{\mathrm{RASP5V}} - U_{\mathrm{SAT}}}{I_{\mathrm{B}} \cdot 3} - R_{\mathrm{OPTO}} = \frac{5 - 0.4}{1,65 \cdot 10^{-3}} - 45 = 2,7 \text{ k}\Omega, \tag{2.6}$$

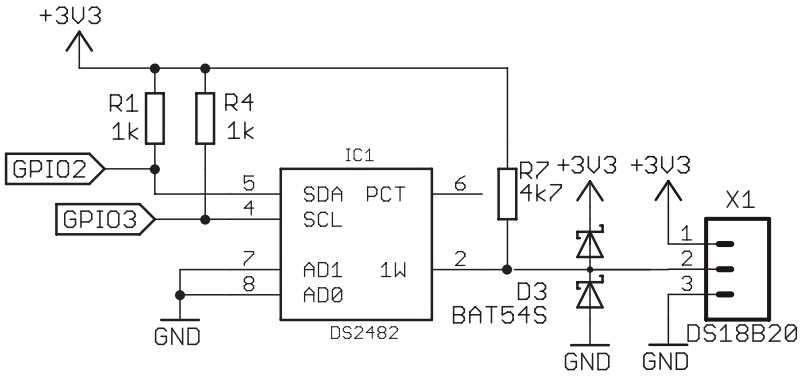
kde U_{SAT} je velikost saturačního napětí tranzistoru, R_{OPTO} odpor optočlenu.

2.3.2 Teplotní čidla

Měření teploty je zajištěno pomocí teplotního čidla DS18B20. Jedná se o digitální teploměr s měřícím rozsahem od –55°C do 135°C, s přesností měření $\pm 0,5$°C. Digitální čidlo vrací přímo hodnotu teploty pomocí sběrnice 1-Wire, kdy je vyžadován pouze jeden datový vodič. Každé teplotní čidlo má jedinečný 64 bitový sériový kód, který umožňuje pracovat s více čidly na stejné sběrnici.

Teplotní čidlo by bylo možné přímo připojit k pinu GPIO a softwarově emulovat komunikaci sběrnice 1-Wire. Toto řešení má však velkou nevýhodu. Připojené čidlo by fungovalo jako dlouhá anténa, která by zachycovala rušení. Sběrnice 1-Wire není v Raspberry implementována hardwarově, ale pouze jako software a to na pinu GPIO 4. Tyto porty jsou přímo připojeny k CPU a rušení by mohlo vést ke zničení celého Raspberry.

Proto je mezi teplotní čidla a Raspberry zapojen integrovaný obvod DS2482. Jedná se o konvertor sběrnic I^2C na 1-Wire, jak je zobrazeno na obr. 2.4. Tímto docílíme odstínění procesoru počítače od časování na sběrnici. Další výhodou je jednoduchá výměna v případě poškození.



Obr. 2.4: Zapojení čidla DS18B20 přes konvertor DS2482

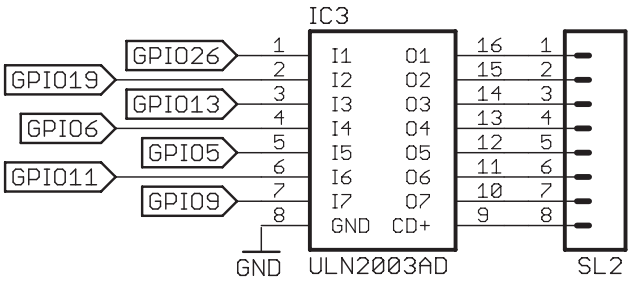
Konvertor je připojen k Raspberry na konkrétní piny GPIO, které umožňují komunikaci po sběrnici I²C. Na sběrnici musí být vysoká logická úroveň signálu v případě, kdy nedochází ke komunikaci. Proto jsou k jednotlivý datovým vodičům připojeny pull up rezistory R1 a R4. Jejich nejmenší hodnota vyplývá z rovnice:

$$R_{\mathrm{R1,R4}} = \frac{U_{\mathrm{RAS33V}} - U_{\mathrm{UBDS24}}}{I_{\mathrm{I^2C}}} = \frac{3,3 - 0,4}{3 \cdot 10^{-3}} = 966\ \Omega, \tag{2.7}$$

kde U_{UBDS24} je úbytek napětí na konvertoru, $I_{\mathrm{I^2C}}$ je proud na sběrnici. To samé platí i pro sběrnici 1-Wire, kde se používá velikost pull-up rezistoru 4,7 kΩ. Podrobnější informace v katalogovém listu [15]. Dále je do obvodu zařazena duální Schottkyho dioda D3, která slouží k omezení napětí na datovém vodiči.

2.3.3 Tranzistorové pole

Ke spínání zátěží s externí hodnotou napětí bude zařízení osazeno Darlingtonovým tranzistorovým polem ULN2003A. Jedná se o strukturu složenou ze dvou bipolárních tranzistorů. Tranzistory jsou připojené za sebou tak, že proud zesílen prvním tranzistorem je dále zesílen druhým. Podrobnější informace v katalogu [16]. Umožňuje spínání induktivních zátěží, které jsou napájené až 50 V. Zapojení je zobrazeno na obr. 2.5.



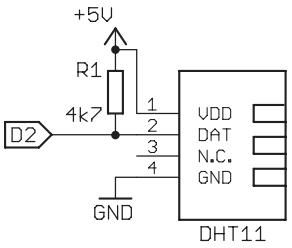
Obr. 2.5: Zapojení Darlightonova tranzistrového pole ULN2003A

2.4 Arduino periferie

Jak již bylo uvedeno v kapitole 2.2, Arduino slouží k obsluze připojených senzorů. Prostřednictvím jeho analogových a digitálních vstupů je zaznamenávána vlhkost okolního prostředí. Dále budou měřeny hodnoty koncentrace CO a CO_2 v daném místě. Základní požadavek byl, aby celkový proud připojených senzorů nepřesáhl 400 mA, aby nebylo nutné použití externího zdroje napětí.

2.4.1 Senzor vlhkosti

Pro snímání vlhkosti byl zvolen senzor DHT11. Jedná se o cenově dostupný, digitální snímač teploty a vlhkosti. Je vybaven digitální technologií sběru a snímání hodnot. Vlhkost měří v rozmezí 20 - 90 % s přesností ±5 %. Dále disponuje vysokou spolehlivostí a dlouhodobou stabilitou. Podrobnější informace v katalogovém listu [17]. Připojení senzoru DHT11 k Arduinu je zobrazeno na obr. 2.6.



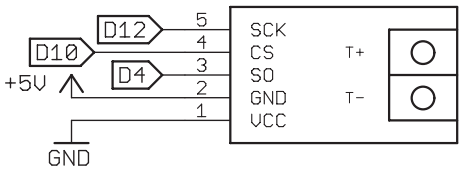
Obr. 2.6: Připojení čídla vlhkosti DHT11 k Arduinu

Senzor je napájen napětím 5 V, připojením na daný pin k Arduinu. Celkový odebíraný proud je 3 mA při měření a ve stand by režimu je uveden odběr 60 µA. Hodnoty vlhkosti jsou již v digitální formě vyčítány mikrokontrolerem, připojením 1-Wire sběrnice k výstupu označeném DATA. Tento datový vodič je přes pull-up rezistor 4,7 kΩ připojen na napájecí napětí.

2.4.2 Senzor teploty

Jako další senzor teploty byl zvolen termočlánek. Jeho výhoda na rozdíl od již zvolených teplotních snímačům je v jeho měřícím rozsahu. Vybraný termočlánek typu K s řídicím obvodem MAX 6675 umožňuje měřit teplotu v rozsahu –200°C až 1300°C, s rozlišitelností 0,25°C a přesností $\pm 1,5$ %. Zapojení termočlánku je zobrazeno na obr. 2.7. Podrobnější informace viz katalogový list [18].

Termočlánek komunikuje pomocí sběrnice SPI, kdy se využívají 3 vodiče. Pomocí jednoho vodiče musí komunikující mikrokontroler generovat hodinový signál. Další vodič má funkci výběr čipu, která je aktivní v logické nule. Poslední vodič je datový výstup.



Obr. 2.7: Připojení termočlánku k Arduinu

2.4.3 Senzory plynu

Cenově dostupné senzory nabízí firma Winsensor. Jedná se o řadu senzorů MQ, která nabízí snímače různých druhů plynu. Tyto senzory využívají zabudovaný ohřívač s elektrochemickým snímačem a používají se v interiérech. Výstup ze senzorů je analogový a lze ho číst analogovým vstupem Arduina.

Oxid uhelnatý

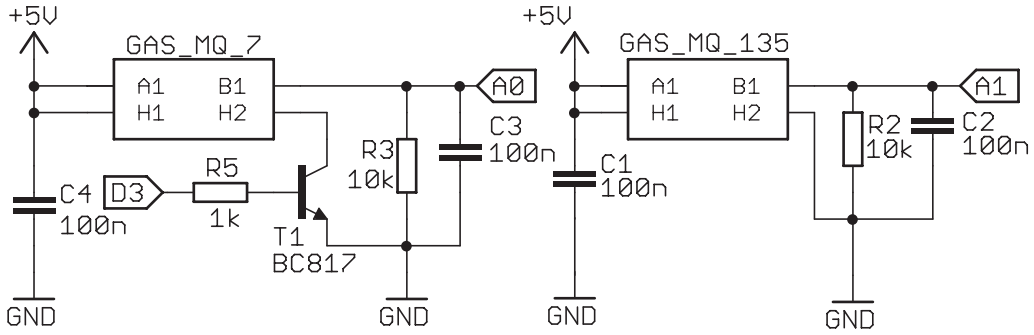
K měření koncentrace oxidu uhelnatého v atmosféře byl vybrán senzor MQ-7. Vyznačuje se velkou citlivostí a dlouhou stabilní životností. Je schopný zaznamenávat koncentraci v rozsahu 20 ppm-2000 ppm. Celkový odebíraný proud by měl být menší než 180 mA. Podrobnější informace katalogový list [19]. Zapojení je zobrazeno na obr. 2.8 vlevo.

Napájení je vedeno přes Arduino výstup 5 V, maximální odebíraný proud 180 mA. Snímač má vnitřní topný článek připojený k pinům H. Vybraný senzor pracuje ve dvou režimech studené a teplé žhavení, které se periodicky opakují. Při teplém žhavení je potřeba senzor napájet napětím 5 V a při studeném 1,4 V. Změna napětí je zajištěna pomocí pulzně šířkové modulace generované na digitálním výstupu D3. Modulací je spínán tranzistor T1, který senzor uzemní.

Oxid uhličitý

Pro měření koncentrace tohoto plynu byl zvolen snímač MQ-135. Tento snímač je citlivý na různé plyny (Benzen, Alkohol, NH_3, CO_2). Prodává se jako snímač kvality vzduchu, ale je ho možné použít ke snímání CO_2. Podrobnější informace manuál [20]. Připojení viz obr. 2.8 vpravo.

Senzor MQ-135 je podobně zapojen jako předchozí senzor. Zde se však žhavicí cykly nemění, senzor je připojen na napájecí napětí 5 V. Hodnota koncentrace je určována z naměřeného výstupního napětí.



Obr. 2.8: Připojení senzorů nebezpečných plynů vlevo MQ-7, vpravo MQ-135

2.4.4 Nezapojené vstupy/výstupy

Pro dodatečné připojení dalších senzorů a výstupů jsou volné piny Arduina opatřeny volnými svorkami. Detailnější popis viz přiložené celkové schéma zapojení A.1.

2.5 Napájení

2.5.1 Požadavky na napájení

Před samotným návrhem napájecího obvodu je třeba určit celkový proud, který bude systém odebírat. V manuálu pro Raspberry je uvedeno doporučení, aby napájecí zdroj dodával minimálně 2,5 A. Tato hodnota je zde uvedena, neboť se počítá s plně proudově vytíženými USB porty. Bez plně zatížených USB portů lze předpokládat spotřebu 1 A. Odebíraný proud pro Raspberry 1 A a 0,5 A pro připojené Arduino pomocí USB. Z připojených periferií budou mít největší proudový odběr reléové výstupy, které budou napájené přímo z 5 V pinu s odebíraným proudem max 0,1 A. Celkový odběr proudu je odhadován asi na 1,7 A, takže výsledný požadavek na napájecí zdroj jsou 2 A.

Monitorovací systém bude sloužit k monitorování parametrů, které jsou důležité k ochraně lidského zdraví a majetku. Proto je tedy nezbytné, aby toto zařízení nebylo závisle pouze na napájení ze sítě. Z tohoto důvodu je zařízení vybaveno záložním zdrojem napětí, kterým jsou v tomto případě nabíjecí baterie. U baterií je velkou nevýhodou proměnlivá hodnota napětí okolo referenční hodnoty. Tudíž baterie nelze připojit přímo k obvodu, ale je potřeba použít napěťového stabilizátoru či DC-DC měniče.

Při použití napěťového stabilizátoru by vznikl problém s jeho vlastním napěťovým úbytkem, nejčastěji 1,5 V a jeho malou účinností. Kdyby byl použit oficiální zdroj k Raspberry, na výstupu by bylo snížené napětí 3,5 V. Pokud bychom na stejné

řešení použili jiného zdroje napětí např. 12 V, stabilizátor by měl velký ztrátový výkon viz rovnice 2.8, který by byl přeměněn na teplo.

$$P = (U_{\mathrm{VST}} - U_{\mathrm{VYST}}) \cdot I_{\mathrm{VYST}} = 7 \cdot 2 = 14\ W. \tag{2.8}$$

Výhodné v tomto případě je použití DC-DC měniče, který se dnes používá v napájecích zdrojích PC. Jeho hlavní výhodou je velká účinnost bez ztrátového tepla.

2.5.2 Baterie

Ve spotřební elektronice se pro své výborné vlastnosti používají baterie typu Li-Ion a Li-Pol.

Baterie Li-Ion jsou nejvíce používaným typem baterií s jmenovitým napětím 3,7 V. Baterie tohoto typu netrpí efektem líné baterie a ani paměťovým efektem. Není nutné před nabíjením plně vybít a kapacita se nesnižuje. Snesou kolem 400-600 nabíjecích cyklů, poté dochází k poklesu kapacity [21].

Li-Pol baterie mají podobné vlastnosti jako výše uvedené Li-Ion, liší se v menších rozměrech. Lze je i formovat a ohýbat a jsou lehčí. Lépe si poradí s mrazem, během času ztrácí kapacitu o něco málo rychleji než Li-Ion, ale to je nepatrné. Jejich velkou nevýhodou je potom vyšší pořizovací cena.

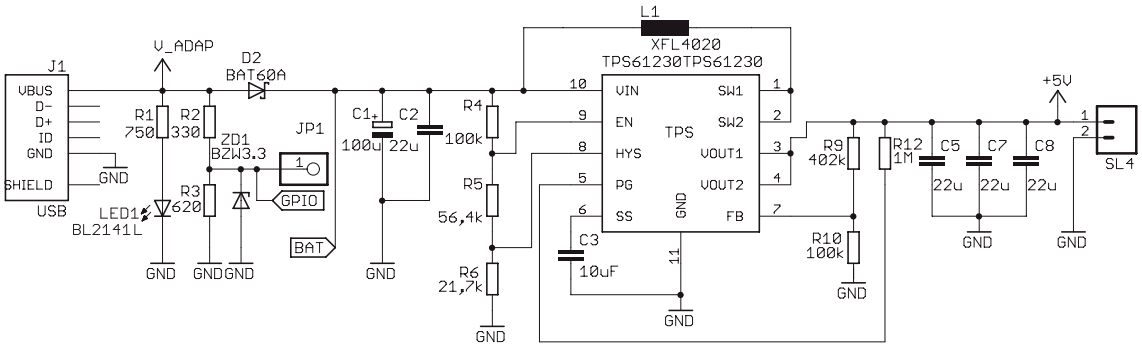
Pro zařízení byl vybrán Li-Pol bateriový článek modelovým označením 854165, zobrazeno na obr. 2.9. Napětí článku je 3,7 V o celkové kapacitě 2500 mAh. Článek je schopný dodávat proud až 4,6 A. Bateriový článek obsahuje ochranný elektronický obvod, který chrání baterii před přepětím a automatický baterii odpojí z obvodu při poklesu napětí pod mezní hodnotu 2,6 V.

Obr. 2.9: Zvolená Li-Pol baterie

2.5.3 Napájecí obvod

Na základě výše uvedených požadavků 2.5.1 a vybrané baterii 2.5.2 byl sestaven obvod pro napájení. Bylo rozhodnuto, že zařízení bude napájeno pomocí originálního adaptéru pro Raspberry s napětím 5 V a dodávaným proudem 2,5 A. Pomocí tohoto zdroje budou dobíjeny i připojené baterie.

Pro regulaci napětí byl zvolen DC-DC zvyšující měnič TPS61230. Jedná se o DC-DC měnič s nastavitelným výstupním napětím 1,8 - 5 V a maximálním výstupním proudem 2,1 A. Velikost vstupního napětí lze měnit předřadnými odpory, což je dobré pro omezení pracovního rozsahu, aby nedocházelo k vybíjení baterií přes stanovenou mez. Obvod je zobrazen na obr. 2.10. Podrobnější informace v katalogovém listu [22].



Obr. 2.10: Napájecí obvod se zvyšujícím DC-DC měničem TSP61230

Hodnota předřadného rezistoru pro diodu LED1 je dána rovnicí 2.2. Tato dioda slouží k indikaci připojeného napětí ze sítě. Podobnou funkci má i odporový dělič tvořený rezistory R2 a R3, který slouží k indikaci síťového napětí pro Raspberry. Hodnoty rezistorů viz rovnice 2.9. Za děličem je připojená Zenerova referenční dioda s napětím 3,3 V pro ochranu portu GPIO.

$$U_2 = U_{\text{adap}} \cdot \frac{R_{\text{R2}}}{R_{\text{R1}} + R_{\text{R2}}} \Rightarrow R_{\text{R3}} = 620\ \Omega \Rightarrow R_{\text{R2}} = 330\ \Omega, \tag{2.9}$$

kde U_2 je maximální napětí pro vstupní GPIO a U_{ADAP} napájecí napětí z adaptéru.

Dioda D2 brání proniknutí napětí z baterie k obvodům zapojených před diodou. Výpočet předřadných odporů pro omezení vstupního napětí je dáno vztahy:

$$U_{\text{IN\_OFF}} = U_{\text{TH\_OFF}} \cdot \left(1 + \frac{R_{\text{R4}}}{R_{\text{R5}} + R_{\text{R6}}}\right), \tag{2.10}$$

$$U_{\text{IN\_ON}} = U_{\text{TH\_ON}} \cdot \left(1 + \frac{R_{\text{R4}}}{R_{\text{R5}}}\right). \tag{2.11}$$

Kde $U_{\mathrm{IN\_ON}}$ je požadované napětí pro spuštění měniče, $U_{\mathrm{TH\_ON}}$ referenční napětí pro spuštění hodnota 1,19 V. Dále $U_{\mathrm{TH\_OFF}}$ je referenční napětí pro vypnutí měniče hodnota 1,14 V a $U_{\mathrm{IN\_OFF}}$ je požadovaná hodnota napětí pro vypnutí měniče.

Výsledné odpory pro spouštěcí napětí $U_{\mathrm{IN\_ON}}$ 3,3 V DC-DC měniče a pro odpojení při napětí menší než 2,6 V $U_{\mathrm{IN\_OFF}}$

$$R_{\mathrm{R4}} = 100\ \mathrm{k\Omega} \Rightarrow R_{\mathrm{R5}} = 56,4\ \mathrm{k\Omega} \Rightarrow R_{\mathrm{R6}} = 21,7\ \mathrm{k\Omega}.$$

Výstupní napětí je dáno odpory R_{R9} a R_{R10} dle vztahu:

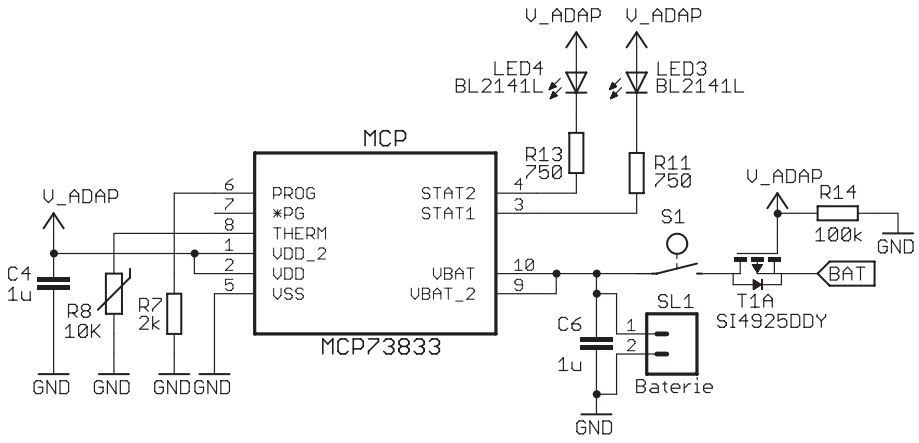
$$U_{\mathrm{OUT}} = U_{\mathrm{FB}} \cdot \left(1 + \frac{R_{\mathrm{R9}}}{R_{\mathrm{R10}}}\right), \tag{2.12}$$

kde U_{FB} je referenční výstupní napětí dle katalogu 1 V. Potom jednotlivé odpory:

$$R_{\mathrm{R9}} = 402\ \mathrm{k\Omega}\ \Rightarrow R_{\mathrm{R10}} = 100\ \mathrm{k\Omega}.$$

2.5.4 Nabíjecí obvod pro baterie

Li-Pol baterie se nabíjí konstantním napětím a proudem. Konstantním proudem se nabíjí až do dosažení mezního napětí, poté se na baterii udržuje napětí a nabíjecí proud samovolně klesá. Provozní teplota nesmí překročit 60°C. Pro nabíjení Li-Pol baterií existují přímo integrované obvody, které obsahují veškerou logiku pro řízení nabíjení. Pro nabíjení vybraného akumulátoru, byl vybrán obvod MCP73833T. Integrovaný obvod obsahuje potřebnou logiku včetně možnosti připojení termistoru pro sledování teploty. Dále je možné připojit signalizační diody pro sledování chodu nabíjení. Podrobnější informace [23]. Schéma zapojení obvodu MCP73833T je zobrazeno na obr. 2.11.



Obr. 2.11: Zapojení nabíjecího obvodu MCP73833T

Na signalizační výstupy je možné připojit LED s proudovým odběrem do 7 mA. Výpočet odporu podobně jako rovnice 2.2. Obvod je přímo napájen z adaptéru 5 V.

Na svorkovnici SL1 je připojen bateriový článek, vývod pokračuje přes tranzistor T1. Pokud je připojeno napětí z adaptéru, tranzistor T1 je uzavřen a proud z baterie přes něj neteče. V opačném případě při odpojení napětí je tranzistor otevřen a připojený obvod je napájen z baterií. Podrobnější informace [24]. Spínač S1 slouží k odpojení napájení z baterie v případě, kdy je vyžadováno úplné vypnutí zařízení. Velikost nabíjecího proudu je dána vztahem:

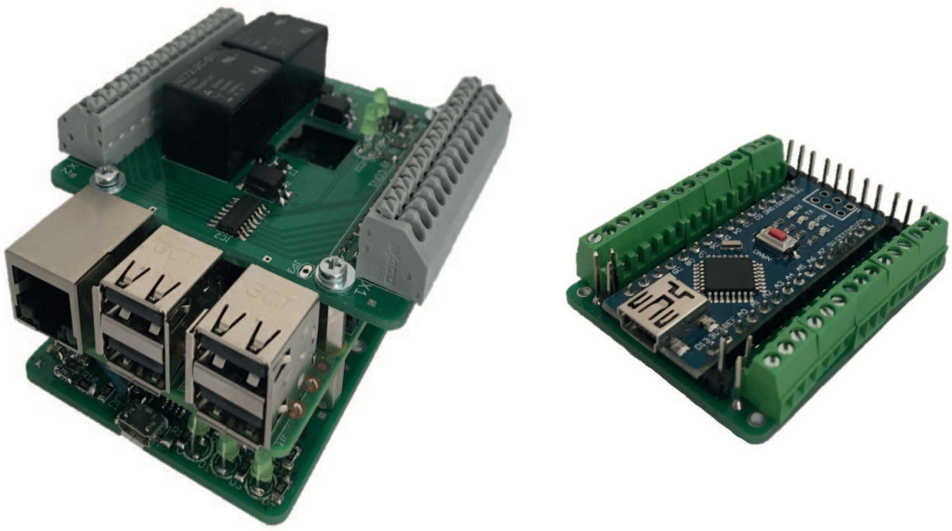
$$I_{\mathrm{REG}} = \frac{1000}{R_{\mathrm{R7}}} = \frac{1000}{2000} = 500 \text{ mA}. \tag{2.13}$$

2.6 Plošné spoje

Dle výše uvedeného návrhu zapojení byly sestaveny plošné spoje, které jsou zobrazeny v přílohách. Nejprve byly v domácím prostředí vyrobeny prototypy navržených desek plošných spojů, na kterých byla ověřena jejich funkčnost.

Po otestování a upravě návrhu byla zadána výroba plošných spojů u čínské firmy Seeed Studio, která nabízí profesionální výrobu plošných spojů za malé finanční náklady.

Vyrobené plošné spoje byly osazeny součástkami a bylo sestaveno celé zařízení. Funkční jednotka Raspberry zobrazena na obr. 2.6 vlevo a Arduino na obr. 2.6 vpravo.



Obr. 2.12: Osazené Raspberry s napájecí deskou vlevo, Arduino vpravo

3 NÁVRH SOFTWARE

Pro oživení zařízení bylo potřeba navrhnout obslužný software. Na Raspberry jsou jednotlivé periferie zařízení ovládány pomocí python skriptu. Nastavení ovládaní systému je zajištěno pomocí webového rozhraní a ovládáno lokálně. Data ze senzorů jsou ukládány do databáze pro možné pozdější čtení naměřených dat. Program pro Arduino je napsán pomocí knihovny Wiring.

3.1 Nastavení Raspberry

Před samotným návrhem programu byla na Raspberry nastavená statická IP adresa. Po instalaci systému Debian je na Rasperry nastavená dynamická IP adresa. To znamená, že se muže měnit při opětovném připojení k PC. Tato situace by vyžadovala po každém zapojení kontrolu, zda nedošlo ke změně IP. Při nastavení statické adresy je tato adresa neměnná, takže vždy bude známa IP adresa vytvořeného systému.

Nastavení statické IP adresy je provedeno změnou nastavení systémového souboru s cestou */etc/dhcpcd.conf*. Na konci souboru je dopsán kód ve tvaru viz 3.1.

Výpis 3.1: Nastavení statické IP adresy

```
interface eth0
static ip_address=192.168.1.104/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

3.2 Teplotní čidla DS18B20

Teplotní čidla jsou připojené k Raspberry přes převodník sběrnice I^2C na 1-Wire pomocí konvertoru DS2482. Pro získání aktuálních hodnot teplot z připojených čidel je potřeba povolit a nastavit komunikaci po sběrnici I^2C. Celý postup je zobrazen ve výpisu kódu 3.2.

Vytvořenou sekvencí kódu je zajištěno, že aktuální teploty z čidel jsou k dispozici v souboru s cestou */mnt/1wire*. V souboru se nacházejí složky, které jsou pojmenovány podle adres připojených čidel. Při otevření souboru *temperature* je možné přečíst aktuální teplotu naměřenou daným čidlem. Pro obsluhu čidel byly vytvořeny dvě funkce pro čtení hodnot a to:

Výpis 3.2: Sekvence příkazu pro nastavení komunikace po sběrnici I^2C

```
sudo apt-get install i2c-tools python-smbus //instalace
sudo i2cdetect -y 1  //ověření funkčnosti
sudo apt-get install owfs ow-shell //instalace owfs jádra
sudo mkdir /mnt/1wire //vytvoření souboru pro tepl. čidla
sudo nano /etc/owfs.conf  //otevření souboru a přidaní řádku:
  device = /dev/i2c-1     //zařízení
  mountpoint = /mnt/1wire //cesta k souboru
  Celsius                 //jednotka
  allow_other
  error_print = 0
  error_level = 0
sudo nano /etc/fuse.conf
  odstranění  u user_allow_other
```

GetSensorsAddress()

Tato funkce vrací adresy připojených čidel. Každé teplotní čidlo DS18B20 má unikátní adresu, která začíná číslovkou 28 a dále následuje několik čísel a znaků. Vytvořená funkce prochází adresáře ve složce připojených čidel a pokud se zde nacházejí složky, které začínají číslem *2* vrátí adresy těchto připojených čidel. Funkce je zobrazena ve výpise 3.3.

ReadTemperature()

Tato vytvořená funkce pracuje se seznamem adres připojených čidel získaným pomocí předchozí funkce *GetSensorsAddress()*. Pro jednotlivé adresy jsou přečteny hodnoty teplot. Jedná se tedy o otevření souboru *temperature* ve složce se jménem příslušného senzoru. Pokud soubor s adresou nebyl nalezen navrací funkce pomlčku. Podrobněji zobrazeno ve zdrojovém kódu 3.3.

Hodnoty z připojených čidel jsou vyčítány a dále ukládány do XML souboru *data*, bude popsáno později. Dále jsou zde také ukládány adresy připojených čidel a časové razítko přečtených teplot. Data v tomto souboru potom slouží pro distribuci dat pro vytvořený webový server.

Výpis 3.3: Získání hodnot teploty z čidel DS18B20

```
def GetSenzorAdress(self):                                          1
  path = "/mnt/1wire"          //cesta k souboru                   2
  dirs = os.listdir(path)      //seznam souborů                    3
  outputdata = ['Error', 'Error', 'Error', 'Error']                4
  // přednastavené hodnoty                                         5
  output = []                  //inicializace výstupu              6
  for file in dirs:            //cyklus pro soubory ve složce      7
    if file[0] == "2":         //indikace připojených čidel        8
      output.append(file)      //přidání prvku                     9
  output=list(set(output))     //smazání zdvojených hodnot         10
  i=0                          //nastavení hodnoty                 11
  for files in output:         //cyklus pro hodnoty output         12
    outputdata[i] = output[i] //naplnění výstupu                   13
    i += 1                     //inkrementace proměnné             14
  return vystupdata            //vrácení hodnoty                   15
                                                                    16
def ReadTemperature(self, ValueAddress):                            17
  address ="/mnt/1wire/"+ ValueAdress +"/temperature"               18
  //cesta k souboru                                                 19
  try:                                                              20
    file = open(address, "r")  //otevření soubrou pro čtení        21
    temperature =  file.read() //přečtení hodnoty                  22
  except IOError:              //pokud chyba                       23
     output.append("-")        //zapiš znak pomlčky                24
  else:                                                             25
     file.close()              //uzavření souboru                  26
     b_temperature = float(temperature)      //konverze            27
     temperature = round(b_temperature, 0)   //zaokrouhlení        28
     temperature = int(temperature)          //konverze            29
     output.append(temperature)              //zapis do pole       30
  return temperature                         //vrácení hodnoty     31
```

3.3 Ovládání výstupů Raspberry

Pro ovládání výstupů, které jsou připojené k Raspberry je potřeba nejprve inicializovat chování GPI pinů. Zda bude daný pin sloužit jako vstup nebo jako výstup, tedy zda je hodnota z něj čtena nebo do něho zapisována.

init()

Jedná se o inicializaci GPIO pinů, nastavení zda daný pin bude vstupní či výstupní a typ značení GPIO pinu. Zobrazeno ve zdrojovém kódu 3.4.

Daný stav je potom nastavován zápisem hodnoty *True* nebo *False* a číslo příslušného pinu např. *GPIO.output(7, True)*.

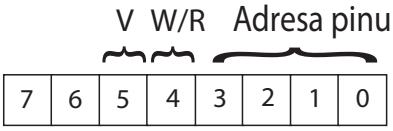
Výpis 3.4: Příklad inicializace GPIO

```
def __init__(self):                                                    1
  GPIO.setmode(GPIO.BCM)    //číslování GPIO dle nazvu                 2
  GPIO.setup(4, GPIO.OUT)   //nastaveni GPIO na výstup                 3
  GPIO.setup(17, GPIO.OUT) //nastaveni GPIO na výstup                  4
  GPIO.setup(9, GPIO.OUT)   //nastaveni GPIO na výstup                 5
  GPIO.setup(11, GPIO.OUT) //nastaveni GPIO na výstup                  6
  GPIO.setup(5, GPIO.OUT)   //nastaveni GPIO na výstup                 7
  GPIO.setup(6, GPIO.OUT)   //nastaveni GPIO na výstup                 8
  GPIO.setup(13, GPIO.OUT) //nastaveni GPIO na výstup                  9
  GPIO.setup(19, GPIO.OUT) //nastaveni GPIO na výstup                  10
  GPIO.setup(26, GPIO.OUT) //nastaveni GPIO na výstup                  11
```

3.4 Arduino

K Arduinu jsou připojeny zvolené senzory, ze kterých je potřeba číst aktuální hodnoty. Jejich připojení je neměnné a nelze dále modifikovat. Dále je možné k Arduinu připojovat na vyvedené piny další senzory popřípadě výstupy. Ty je možné ovládat bez modifikace programu vytvořeného pro obsluhu Arduina. Proto je vytvořen program, který má v sobě zahrnuty obě varianty pro připojení výstupního nebo vstupního zařízení na nezapojené piny.

Arduino komunikuje s Raspberry pomocí sériové sběrnice. Data jsou zasílána přímo přes propojovací USB datový kabel. Pro komunikaci je vytvořen vlastní jednoduchý komunikační protokol. Zařízení si mezi sebou zasílají bajtové zprávy. Raspberry je v roli Master, generuje příkazy a Arduino má přidělenou roli Slave, na zaslané požadavky odpovídá. Základní struktura komunikačního bajtu je zobrazena na obr. 3.1.



Obr. 3.1: Zobrazení bajtu s popisem funkce jednotlivých bitů

První 4 bity (označené 0 - 3) z komunikačního bajtu nesou adresu konkrétního pinu. Pátý bit určuje, zda se bude chovat jako vstup nebo výstup. Pokud je zvolen pin jako výstupní, 6. bitem se určuje hodnota výstupu tedy logická jednička či nula.

Nevýhodou komunikace po sériové lince je možná ztráta dat nebo chybějící část zprávy. Proto je vhodné komunikaci po sériové sběrnici vybavit mechanismem kontroly přijatých dat. Mezi používané metody patří například:

- Parita
- Modulo
- Cyklický redundantní součet (CRC)

Vzhledem k tomu, že je zasíláno pouze malé množství dat. Je pro zajištění správně přijatých dat vybrán mechanismus úplné kopie zprávy. Pokud při přenosu dat dojde k chybě, přijaté zprávy se budou lišit.

Pro obsluhu sériové linky je na straně Raspberry vytvořena funkce ***SendSerialCmd()***, tato funkce je zobrazena ve výpise 3.5.

Výpis 3.5: Vytvořená funkce pro seriovou komunikaci

```
def SendSerialCmd(self, address, cmd):                              1
  address = str(address)      //převod parametru na string        2
  cmd = chr(cmd)              //převod zprávy na znak             3
  ser = serial.Serial('/dev/'+address, 9600)                        4
  //konfigurace komunikace                                          5
  dataT1=0                              //pocatecni hodnota         6
  dataT2=0                              //pocatecni hodnota         7
  loop=0                                //pocatecni hodnota         8
  while loop <3:                        //opakuj 3x                 9
    ser.write(cmd)                      //zaslání zprávy            10
    time.sleep(0.5)                     //zpoždění                  11
    while ser.inWaiting() > 0:          //pokud data                12
        dataT1 = ser.readline()         //zápis přijatých dat       13
    if dataT1!=0:                       //pokud přijata zpráva      14
        time.sleep(0.4)                 //zpoždění                  15
        while ser.inWaiting() > 0:      //pokud data                16
            dataT2 = ser.readline()     //zápis přijatých dat       17
    if dataT2!=0:                       //pokud přijata zpráva      18
        if dataT1==dataT2:              //pokud data stejná         19
           break                        //přerušení while           20
    loop+=1                             //inkrementace loop         21
  ser.close()                           //uzavření ser. komunikace  22
  return dataT2                         //navracení dat             23
```

Arduino na přijatá data reaguje a pokud se jedná o dotaz na hodnotu vstupu, odpovídá odesláním příslušné hodnoty, zdrojový kód zobrazen ve výpise 3.6.

Výpis 3.6: Nastavení chování pinu

```
void setup() {                                                          1
Serial.begin(9600);     //otevření sériového kanálu                     2
  ...                                                                   3
  }                                                                     4
void loop(){                                                            5
  if(Serial.available()>0){                                             6
    DataSerialRead=Serial.read();  //čtení přichozího bajtu             7
    int AddressReceive=DataSerialRead & 15;//maskování bitů             8
    int WriteReadReceive=DataSerialRead & 16;//hodnota 5. bitu          9
    int CommandReceive=DataSerialRead & 32;//hodnota 6. bitu            10
    ...                                                                 11
  }                                                                     12
  switch (AddressReceive){            // přepinač dle adresy            13
    case 3:if(WriteReadReceive==16){//pokud WriteReadReceive=1          14
             pinMode(A2,OUTPUT);      //definování výstup               15
             if (CommandReceive==32){ //pokud CommandReceive=1          16
               digitalWrite(A2,HIGH);}//zapiš log.1 na vystup           17
             else{                    //pokud CommandReceive=0          18
               digitalWrite(A2,LOW);} //zapiš log.1 na vystup           19
               }                                                        20
            else{                 // pokud WriteReadReceive!=1          21
             pinMode(A2,INPUT);  //definováni pinu jako vstup           22
             float Output2=analogRead(A2);//čtení vstupu                23
             Output2=Output2*5.0/1023.0;  //hodnota napětí             24
             Serial.println(Output2);     //zaslání odpovědi            25
             delay(500);                  //zpoždění                    26
             Serial.println(Output2);     //opětovné zaslání            27
                      }                                                 28
                 break;                                                 29
    ...                                                                 30
  }                                                                     31
}                                                                       32
```

Nejprve je inicializována sériová linka s přenosovou rychlostí 9600 bitů za sekundu. V hlavní větvi programu je potom vyhodnocováno zda jsou přijatá data. Při přijetí dat jsou nejdříve extrahovány příslušné bity logickým součinem a uloženy do příslušných proměnných. Poté je na základě přijaté adresy a hodnotách dalších bitů, provedena požadovaná akce. Tedy zápis dat na výstup nebo vrácení hodnoty vstupu.

3.4.1 Senzory plynů CO a CO_2

U zvoleného senzoru oxidu uhelnatého MQ-7 je potřeba přepínat mezi teplým a studeným žhavením. Při teplém žhavení je na vyhřívacích pinech H přivedeno napětí 5 V. Z citlivé destičky se vypaří zbytky po aktuální hodnotě, tato úroveň napětí zde musí působit 60 sekund. Potom by mělo následovat studené žhavení. Kdy na vyhřívacích pinech je přivedena hodnota napětí 1,4 V po dobu 90 sekund a potom následuje vyčtení aktuální hodnoty odporu daného senzoru. Podrobnější informace o průběhu žhavících cyklů jsou popsáné v manuálu [19].

Hodnota 1,4 V je zajištěna pomocí pulzně šířkové modulace PWM. Pro generování PWM je použit pin Arduina označen jako D3. Jedná se o výstup z interního čítače/časovače 2, s jeho pomocí je možné generovat rychlou pulzně šířkovou modulaci (Fast PWM). Hodnota 5 V je získána potom pří 100 % střídě generovaného signálu. Pro studené žhavení, kde je potřeba 1,4 V, získáme hodnotu napětí dle vztahu uvedeného v rovnici 3.1. Nastavení generátoru FPWM viz přiložený kód 3.7.

$$S_{\mathrm{PWM}} = \left(\frac{U_{ef}}{U_{MAX}}\right)^2 \cdot 100 = \left(\frac{1,4}{5}\right)^2 \cdot 100 = 8\% \tag{3.1}$$

Pro obsluhu senzoru oxidu uhličitého MQ-135 není třeba střídání různých vyhřívacích cyklů. Žhavení je vyžadováno po celou dobu měření při konstantní hodnotě napájecího napětí 5 V.

Výpis 3.7: Vytvořený generátor PWM

```
void setTimer2PWM(byte chB) //PWM na pinu D3                    1
{TCCR2A = 0b10100011;       //generování rychlé PWM             2
 TCCR2B = 0b100;            //předdělička 64 pro 976Hz          3
 OCR2A = 0;                 //hodnota střídy                    4
 OCR2B = chB;}              //hodnota střídy                    5
```

U plynového senzoru MQ-7 je tedy potřeba přepínat mezi režimy trvajícími 60 s a 90 s . Pro obsluhu těchto cyklů je využito funkce čítače/časovače 1 s možností vyvolání přerušení po přetečení. Je vytvořeno přerušení po uplynutí 1 s. Při každém vyvolání přerušení je inkrementován obsah proměnné *time*. Hodnota zmíněné proměnné udává uplynutý čas. Tento čas je poté vyhodnocován s režimem, který má následovat. Při zapojení Arduina začne vykonávání teplého vyhřívání a poté studeného vyhřívání, tyto stavy se periodický opakují. Na konci studeného vyhřívání následuje čtení aktuální hodnoty odporu. Podrobněji viz přiložený kód 3.8.

Výpis 3.8: Ovládání přepínání cyklu vyhřívání

```
 TCCR1A=0;                          //č/č 1 v normálním modu          1
 TCCR1B |=0b00000101;               //nastavení předděličky 128       2
 TCNT1=49911;                       //počáteční hodnota pro čítač     3
 TIMSK1 |=(1<<TOIE1);               //povolení č/č 1                  4
 interrupts();                      //povolení přerušení              5
                                                                      6
ISR(TIMER1_OVF_vect)                //vektor přerušení                7
{TCNT1=49911;                       //počáteční hodnota pro čítač     8
time=time+1;                        //čítač přerušeni                 9
 if (time==60 && phase==0){         //po 60 s. a phase 0              10
  setTimer2PWM(21);                 //změna střídy 8\% z 255          11
  time=0;                           //nulování počtu přerušení        12
  phase=1;                          //změna phase                     13
 }                                                                    14
 else if(time==90 && phase==1){//po 90 s. a phase 0                   15
 setTimer2PWM(255);                 //změna střídy 100\%              16
 time=0;                            //nulování počtu přerušení        17
 phase=0;                           //změna phase                     18
 ValueA0=analogRead(A0);            //čtení hodnoty vstupu A0         19
 VrlCO=5.0/1023*ValueA0;}}          //přepočet na napětí              20
```

Pro získání hodnot koncentrace v jednotkách částic na jeden milion, tedy ppm je nutné senzory kalibrovat. Kalibrace se provádí pomocí známé hodnoty koncentrace a grafu závislosti poměrů odporů R_S/R_O na koncentraci daného plynu. Hodnota R_S je dána vztahem:

$$R_S = \frac{(V_{cc} - V_{RL}) \cdot R_L}{V_{RL}}, \tag{3.2}$$

kde R_S je odpor senzoru závislý na koncentraci, R_L je odpor předřadného odporu, V_{RL} je naměřené napětí na odporu R_L,V_{cc} je napájecí napětí senzoru.

Dále je potřeba stanovit hodnotu R_O, která udává hodnotu odporu pro daný snímač. Tato hodnota se stanoví z výpočtu z naměřeného odporu R_S při známe koncentraci.

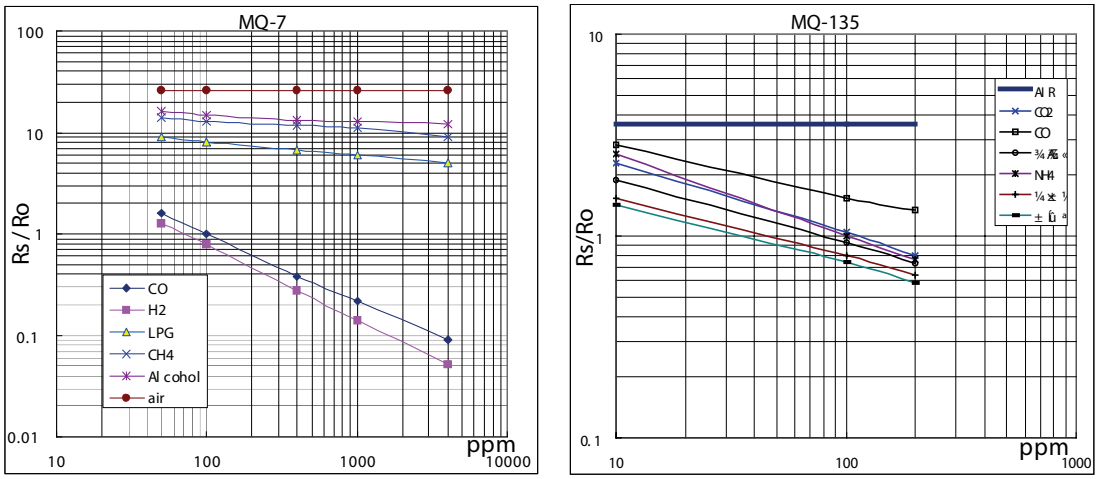
Nejprve je nutné získat závislost odporů na koncentraci pro daný plyn např. v lineárním měřítku. Z charakteristik udávaných výrobcem, které jsou zobrazeny na obr. 3.2, jsou extrahovány hodnoty. Takto získaná data jsou proložena mocninou spojnicí tendru. A jsou získány potřebné závislosti, jak je zobrazeno na obr. 3.3.

Pro plyn CO je přepočet poměru odporu R_S/R_O na koncentraci plynu je roven vztahu:

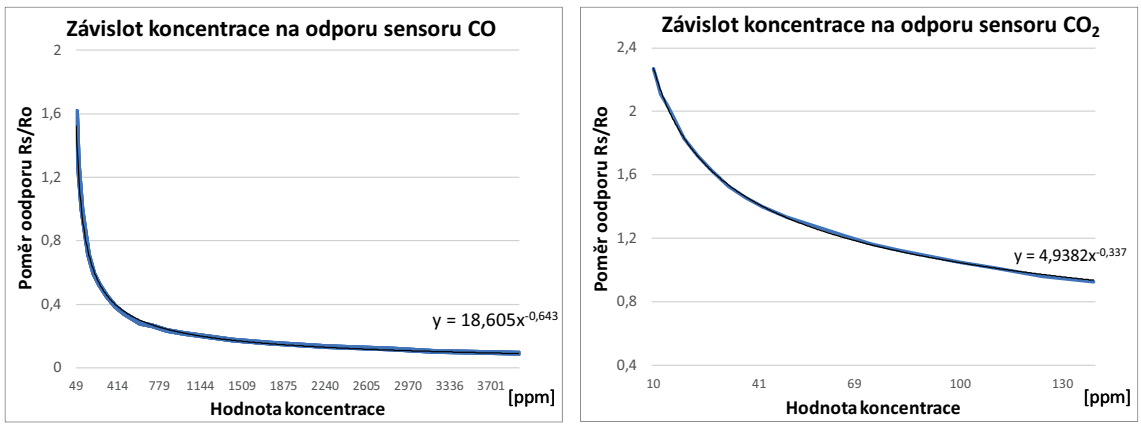
$$\frac{R_S}{R_O} = 18,605 \cdot K_{ppm}^{-0,643} \Rightarrow K_{ppm} = 94,3063 \cdot \left(\frac{R_S}{R_O}\right)^{-1,555}. \quad (3.3)$$

Pro plyn CO2 *je* *přepočet* *poměr* *u* *odporu* R_S/R_O na koncentraci plynu je roven vztahu:

$$\frac{R_S}{R_O} = 4,9382 \cdot K_{ppm}^{-0,337} \Rightarrow K_{ppm} = 114,306 \cdot \left(\frac{R_S}{R_O}\right)^{-2,967}. \quad (3.4)$$



Obr. 3.2: Závislosti poměrů naměřených odporů na koncentraci plynů vlevo pro CO, vpravo pro CO_2 udávané výrobcem [19] a [20]



Obr. 3.3: Získané závislosti poměrů naměřených odporů na koncentraci plynů vlevo pro CO, vpravo pro CO_2 v lineárních souřadnicích

Kalibrace senzoru byla provedena pomocí přístrojů, které měří hodnoty koncentrace daných veličin v jednotkách ppm. Pro senzor oxidu uhelnatého byl zvolen měřící přístroj SFT-111-LCD, jedná se detektor plynu, zobrazující aktuální hodnotu koncentrace oxidu uhelnatého, zobrazeno na obr. 3.4 vlevo. Pro kalibraci senzoru oxidu uhličitého byl použit dostupný přístroj TFA Aircontrol 3000, který je zobrazen na obr. 3.4 vpravo.

Obr. 3.4: Použité kalibrační přístroje, vlevo zobrazen detektror SFT-111-LCD, vpravo TFA Aircontrol 3000

Před kalibrací byly podle instrukcí senzory ponechaný v nepřetržitém provozu po dobu 48 hodin. Poté byly spolu s kalibračním měřícím zařízením umístěny do boxu s uzavíratelným víkem. Poté byly stanoveny hodnoty odporů R_O pro jednotlivé senzory podle vztahů uvedených v rovnicích 3.3 a 3.4.

3.4.2 Sensor teploty a vlhkosti DHT11

Pro čtení hodnot teploty a koncentrace vlhkosti z připojeného čidla DHT11 je použita již vytvořená knihovna. Tato knihovna je volně dostupná na webu. Kód pro základní obsluhu senzoru je zobrazen ve výpise 3.9.

Výpis 3.9: Obsluha senzoru DHT11

```
#include "DHT.h"                    //připojení knihovny                    1
#define pinDHT 2                    //pin s připojeným DHT senzorem         2
#define typDHT11 DHT11              //typ čidla                             3
DHT mojeDHT(pinDHT, typDHT11); //inicializace DHT senzoru                   4
void setup() {mojeDHT.begin(); //povolení komunikace s DHT                  5
}                                                                           6
void loop() {                                                               7
  float tep = mojeDHT.readTemperature();//hodnota teploty                   8
  float vlh = mojeDHT.readHumidity();   //hodnota vlhkosti                  9
}                                                                           10
```

3.4.3 Termočlánek MAX 6675

Stejně jako pro senzor DHT11 je pro termočlánek importována již vytvořená knihovna v tomto případě se jedná o knihovnu *MAX6675*, která je také volně dostupná na internetu. Základní obslužný program pro získání hodnoty teploty podrobněji viz 3.10.

Výpis 3.10: Obsluha senzoru MAX6675

```
#include "max6675.h"                          //připojení knihovny
int pinSO  = 4;                               //nastaveni pinu SO
int pinCS  = 10;                              //nastaveni pinu CS
int pinSCK = 12;                              //nastaveni pinu SCK
MAX6675 termoclanek(pinSCK, pinCS, pinSO);
//vytvoření instance z knihovny
}
void loop(){
float teplotaC = termoclanek.readCelsius(); //naměřená hodnota
}
```

Kódy uvedené ve výpisech 3.10 a 3.9 slouží pro základní obsluhu připojených čidel. Ve vytvořeném programu pro obsluhu Arduina se proměnné s naměřenou hodnotou nachází v cyklu přepínače pro konkrétní pin. Pro tyto piny neplatí možnost změny jejich funkce na výstup.

3.5 Webové rozhraní

3.5.1 Základní koncepce

Pro obsluhu webového rozhraní je vhodné mít shromážděny data, se kterými systém pracuje, na jednom místě. Pro kompletaci dat je vytvořeno několik souborů ve formátu XML (eXtensible Markup Language). Jedná se o rozšířený značkovací jazyk, který je určen pro výměnu dat mezi aplikacemi a pro publikování dokumentů. Pro webové rozhraní jsou vytvořeny následující soubory:

data

V tomto souboru jsou uloženy data, která jsou aktuálně získávána z připojených teplotních senzorů. Základní informaci zde tvoří adresa aktuálně připojených měřících zařízení DS18B20. Tato adresa je klíčová pro čtení aktuální hodnoty teploty a zapisování časového razítka, které informuje o posledním čase přijatých dat. Dále je zde ukládán přidělený název pro snadnou orientaci mezi připojenými senzory. Základní struktura je zobrazená ve výpise 3.11. Každému prvku je přiřazeno identifikační číslo *id*. Podle toho prvku se orientuji naprogramované funkce.

Výpis 3.11: Struktura XML souboru data

```
<temperature id="1">                                          1
  <name>Teplota Vody</name>           //přidělený název     2
  <value>25</value>                   //aktuální hodnota    3
  <address>28.FFF4A0011703</address> //adresa                4
  <time>24.04.2018 14:07</time>       //čas přijatých dat   5
</temperature>                                                6
```

data_output

Zde jsou uložena data potřebná pro obsluhu výstupních zařízení zobrazeno ve výpise 3.12. Element *permit* podává informaci o aktivitě daného výstupu. Položky *threshold*, *hysteresis*, *negation*, jak názvy napovídají, slouží určení aktuální stavu podle výstupu. Údaje *sensorValue* a *sensorAddressValue* podávají zprávu na jaký senzor daný výstup reaguje a jakou má aktuální hodnotu sledované veličiny. Do položky *state* je ukládána informace v jakém stavu se nachází výstup.

Arduino

Poslední XML soubor je vytvořen pro sběr informaci o připojeném Arduinu. Základní struktura je zobrazena ve výpise 3.13 a je velmi obdobná předchozímu souboru. Nachází se zde element pro uchování informace o adrese připojeného Arduina *connection*. Tato adresa se mění při přepojení Arduina. Dále položka *sensorUnit* obsahuje informaci o jednotce v případě připojení vstupního senzoru. Tag *input* určuje zda se jedná o vstup či výstup.

Výpis 3.12: Struktura XML souboru data_out

```
<output id="1">                                                    1
  <sensorValue>25</sensorValue>                                    2
  //aktuální hodnota obsluhovaného vstupu                          3
  <sensorAddressValue>28.FFF4A0011703</sensorAddressValue>          4
  //adresa obsluhovaného vstupu                                    5
  <threshold>25</threshold>      //hodnota aktivace                6
  <hysteresis>5</hysteresis>     //hystereze                       7
  <permit>1</permit>             //povolení                        8
  <negation>0</negation>         //negace                          9
  <state>0</state>               //aktuální stav                   10
  <name>Ventilator</name>        //přidělený název                 11
</output>                                                          12
```

Výpis 3.13: Struktura XML souboru Arduino

```
<inputoutput id="co">                                              1
    <connection>ttyUSB0</connection>  //adresa portu Arduina      2
</inputoutput>                                                     3
<inputoutput id="A2">                                              4
  <input>1</input>                    //chování pinu             5
  <permit>1</permit>                  //aktivita                 6
  <name>CO</name>                     //přidělený název          7
  <threshold>35</threshold>           //hodnota aktivace         8
  <hysteresis>2</hysteresis>          //hodnota hystereze        9
  <negation>0</negation>              //hodnota negace           10
  <addressSensorValue>28.FFF4A0011703</addressSensorValue>          11
  //adresa obsluhovaného vstupu                                    12
  <sensorValue>20</sensorValue>                                    13
  //aktuální hodnota obsluhovaného vstupu                          14
  <sensorUnit>ppm</sensorUnit>        //jednotka                 15
  <state>0</state>                    //aktuální stav            16
</inputoutput>                                                     17
```

3.5.2 Vlastní návrh webového rozhraní

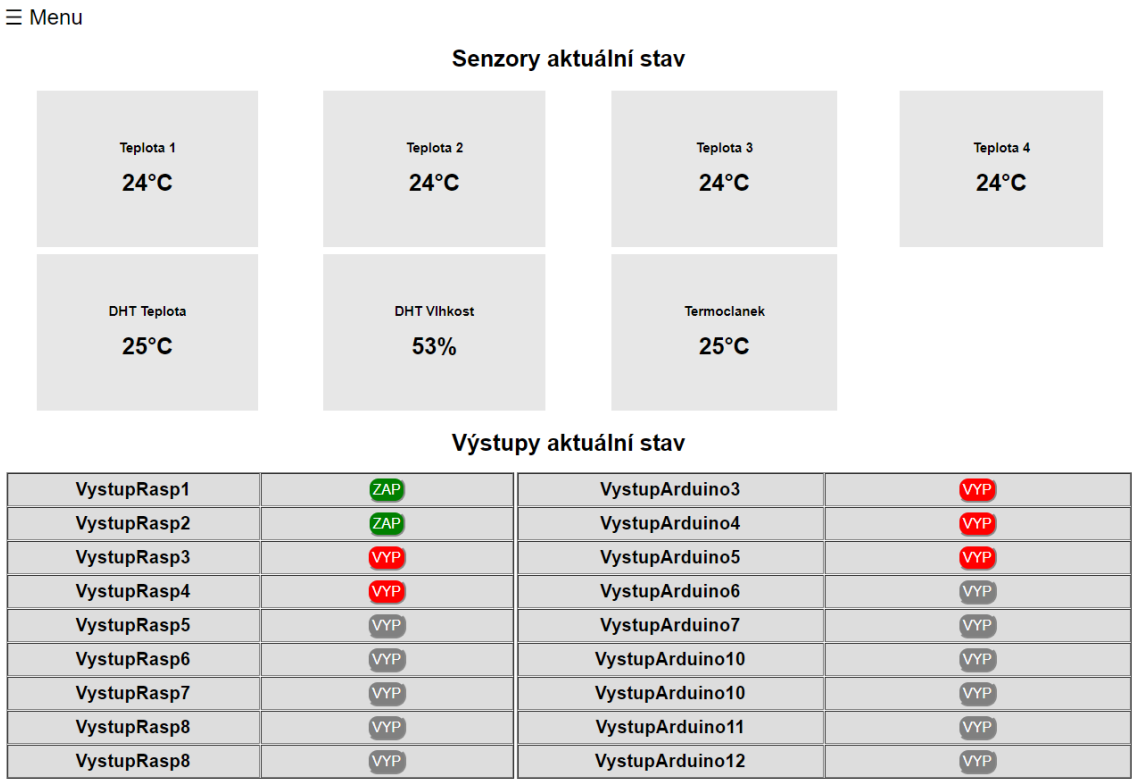
Webové stránky byly vytvořeny pomocí html příkazů, kaskádových stylů css a dynamické informace jsou naprogramovány pomocí JavaScriptu. Webové rozhraní tvoří tyto stránky:

mainpanel

Na této stránce je zobrazen hlavní panel. Z panelu je možné vyčíst aktuální informace ze senzorů a aktivované výstupy. Panel je zobrazen na obr. 3.5 a tvoří ho dvě základní sekce.

Jedna je **Senzory aktuální stav**, kde jsou zobrazeny aktivní senzory a jejich aktuální hodnota. Ta se dynamicky mění bez toho abychom museli aktualizovat webovou stránku. Název teplotních senzorů je nastaven po aktualizaci stránky a dále neměnný. První 4 objekty pro senzory jsou zapnuty trvalé. Pokud je připojeno Arduino a je povolena obsluha jeho vstupních senzorů, mění se počet těchto objektů podle zvolených senzorů.

Druhou část stránky potom tvoří **Výstupy aktuální stav**, kde jsou zobrazeny dostupné výstupy. Je zde zobrazen název příslušného výstupu a jeho aktuální stav. Pokud výstup je neaktivní má indikátor ZAP/VYP šedou barvu. Pokud je daný výstup aktivní, indikátor změní barvu podle aktuální hodnoty výstupu, na červenou při vypnutí a zelenou při zapnutí.

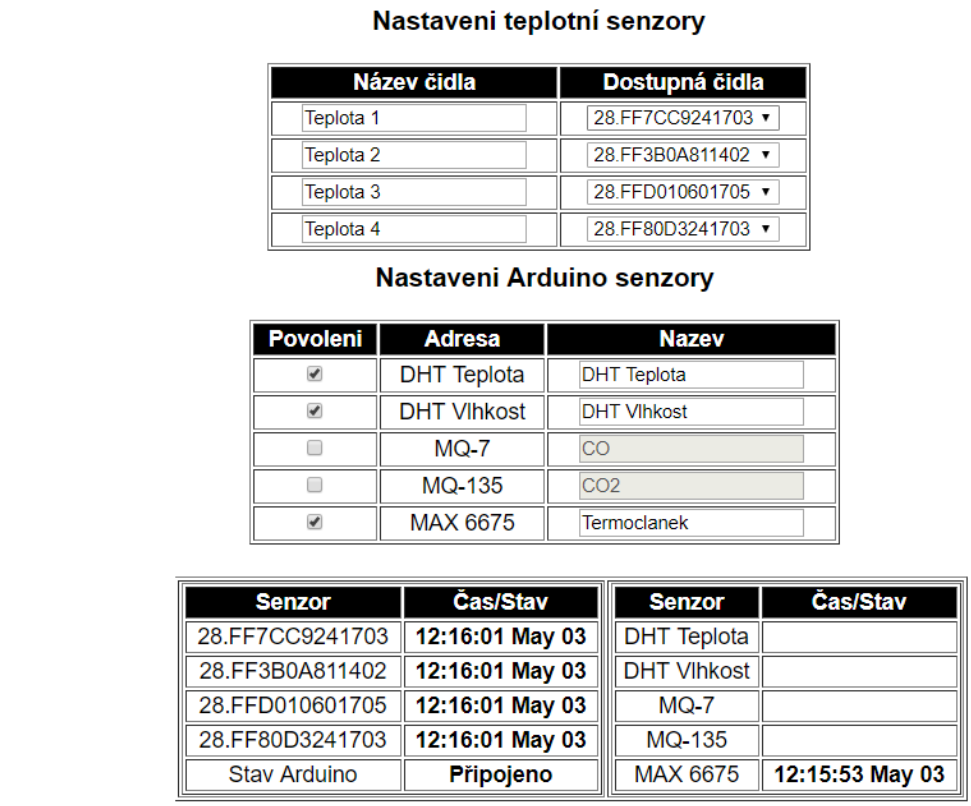


Obr. 3.5: Pohled na vytvořený hlavní panel

settings

Pomocí této stránky uživatel nastavuje funkci celého systémů. Aktivní senzory, výstupy a jejich informace jsou po aktualizaci k dispozici na hlavním panelu. Jsou zde informace o dostupných senzorech a výstupech. Stránka obsahuje pět tabulek, které jsou rozděleny do tří základních oblastí

První z nich tvoří **Nastavení vstupů**, jak je zobrazeno na obr. 3.6. Je zde tabulka, kde se nastavuje název dostupných teplotních senzorů. Předpokládá se, že budou vždy připojeny čtyři senzory, takže je není možné deaktivovat, je pouze možné přidat k danému objektu na panel senzor s konkrétní adresou. Adresy jsou aktualizovány po obnovení stránky, takže jsou zde adresy právě dostupných čidel. Dále v této sekci je tabulka vstupů Arduino. Měnit její parametry lze pouze, když je Arduino připojeno. Pomocí JavaScriptu je dynamicky vyhodnocován stav bez nutnosti obnovení stránky. V poslední tabulce jsou zobrazeny informace o připojených senzorech a je zde zobrazen čas poslední komunikace s daným senzorem, obsluhováno taktéž dynamicky.



Obr. 3.6: Nastavení vstupů

Druhou oblastí je **Nastavení Výstupů**, jak je zobrazeno na obr. 3.7. Zde je možné aktivovat potřebný výstup. Po aktivaci je potřeba nastavit na který z aktivních vstupů má daný výstup reagovat. Dále se zde nastavuje hodnota pro aktivaci,

při jaké úrovni má dojít k aktivaci. Následuje nastavení hystereze, která ovlivňuje aby nedocházelo k zakmitávání výstupu. Posledním nastavovacím prvkem je poté negace. Tento údaj určuje, zda se daný výstup při aktivaci rozepíná či spíná. Dále je zde test výstupu, pokud výstup není aktivní, je možné vyzkoušet funkci ovládání výstupu kliknutím na konkrétní tlačítko.

Povolení	Adresa	Vstup	Název akčního členu
☑	Rele1	28.FF7CC9241703	VystupRasp1
☑	Rele2	28.FF7CC9241703	VystupRasp2
☑	Uln1	28.FF7CC9241703	VystupRasp3
☑	Uln2	28.FFD010601705	VystupRasp4
☐	Uln3		VystupRasp5
☐	Uln4		VystupRasp6
☐	Uln5		VystupRasp7
☐	Uln6		VystupRasp8
☐	Uln7		VystupRasp9

Hodnota aktivace	Hystereze	Negace	Test výstupu
0	5	☐	ON OFF
0	5	☐	ON OFF
28	0	☐	ON OFF
28	0	☐	ON OFF
0	0	☐	ON OFF
0	0	☐	ON OFF
0	0	☐	ON OFF
0	0	☐	ON OFF
0	0	☐	ON OFF

Obr. 3.7: Nastavení výstupů z důvodů zachování čitelnosti rozděleno na dvě tabulky

Poslední částí je **Nastavení Arduino**, náhled zobrazen na obr. 3.8. Základní princip je stejný jako u nastavování výstupu. U Arduina je navíc možné nastavovat vstupy i výstupy. Po aktivování daného pinu je potřeba zvolit zda se jedná o vstup nebo výstup, v případě vstupu je možné nastavit jednotku, která bude zobrazována na hlavním panelu.

Povolení	Adresa	Vstup/Výstup	Reakce	Jednotka
☐	A2	Výstup		C
☐	A3	Výstup		C
☐	A4	Výstup		C
☐	A5	Výstup		None
☐	A6	Výstup		None
☐	D5	Výstup		None
☐	D6	Výstup		None
☐	D7	Výstup		None
☐	D8	Výstup		None
☐	D9	Vstup		None

Název	Hodnota aktivace	Hystereze	Negace	Test výstupu
VystupArduino3	35	2	☐	ON OFF
VystupArduino4	35	0	☐	ON OFF
VystupArduino5	35	0	☐	ON OFF
VystupArduino6	0	0	☐	ON OFF
VystupArduino7	0	0	☐	ON OFF
VystupArduino10	0	0	☐	ON OFF
VystupArduino10	0	0	☐	ON OFF
VystupArduino11	20	3	☐	ON OFF
VystupArduino12	20	5	☐	ON OFF
VystupArduino13	20	5	☐	ON OFF

Obr. 3.8: Nastavení vstupů/výstupů Arduino z důvodů zachování čitelnosti rozděleno na dvě tabulky

3.5.3 Oživení webového rozhraní

Protože jsou veškeré ovládací funkce napsané v python skriptu je výhodné komunikaci s webovým rozhraním řešit stejným vývojovým prostředím, a tím bude umožněno volání vytvořených funkcí. Na internetu je k dispozicí velké množství webových frameworku, které lze programovat pomocí python skriptu. Pro návrh webového rozhraní byl zvolen framework s názvem *Flask*. Jedná se o jednoduchý webový mikroframework pro vytvoření webových aplikacích instalace je zobrazena ve výpise 3.14.

Výpis 3.14: Instalace frameworku flask

```
sudo apt-get install python-pip //instalace modulu pip
sudo pip install flask          //instalace flask
```

Pomocí uvedeného frameworku byl vytvořen management mezi jednotlivými vytvořenými html stránkami a python skriptem. Základní kód viz 3.15.

Výpis 3.15: Zakládní koncepce webového frame worku

```
from flask import Flask //import knihovny flask
app = Flask(__name__)   //inicializace

@app.route('/')
def mainpanel():
  return render_template('mainpanel.html')
  //vrať html stránku mainpanel

@app.route('/settings')
def settings():
  return render_template('settings.html')
  //vrať html stránku setting

if __name__ == '__main__':
app.run(host='0.0.0.0', debug=True)
// běh na http://localhost/
```

Při aktualizací stránek jsou předávána potřebná data z vytvořených xml souborů. Pro tyto účely je vytvořena python funkce:

GetOutXMLFile()

Funkce vrací požadované hodnoty z xml souboru. Tato funkce je volána se třemi základními parametry. *SubEleName*, který určuje název elementů v xml struktuře. Druhý je *index*, který definuje id daného prvku. A poslední je *file*, který určuje z jakého souboru se budou data číst. Celá funkce zobrazena ve výpise 3.16. Nejprve je nastavena cesta k souboru *xml_file*. Při čtení dat je načtena celá struktura xml souboru a poté je přezvat konkrétní obsah. Jedná se to takzvané parsování. Pokud došlo k načtení obsahu souboru pokračuje se získáním konkrétní hodnoty dle id. Pokud nedošlo k naparsování, cyklus se opakuje.

Výpis 3.16: Funkce čtení dat z XML souboru

```
def GetOutXMLFile(self, SubEleName, index, file):                    1
  base_path = os.path.dirname(os.path.realpath(__file__))            2
  xml_file = os.path.join(base_path, "templates/"+file+".xml")       3
  //sestavení cesty k souboru                                        4
  output=0                          //dinicializace proměné output  5
  test=0                            //inicializace proměné test     6
  while test==0:                    //cyklus otevření souboru       7
    try:                                                             8
       tree = et.parse(xml_file)//otevření souboru                   9
       test = 1                                                      10
    except:                                                          11
       test = 0                                                      12
  root = tree.getroot()             //naparsování obsahu            13
  SubEleName = str(SubEleName)      //konverze                      14
  index = str(index)                //konverze                      15
  cesta = ".*[@id='"+index+"']/"+SubEleName                          16
  //sestaveni dotazu id                                              17
  elems = root.findall(cesta)       //nalezení všecho dat           18
  for elem in elems:                                                 19
       output = elem.text           //zapis dat na výstup           20
  return output                                                      21
```

Data získaná popsanou funkcí jsou uloženy do proměnných, se kterými dále pracuje html stránka. Ukázka vrácení hodnot při obnovení stránky je zobrazena ve výpise 3.17.

Výpis 3.17: Vložení dat pro html stránku

```
@app.mainpanel('/')                                                  1
def home():                                                          2
  NameTempSensor1 = pi_control.GetOutXMLFile("name",1,"data")        3
  NameTempSensor2 = pi_control.GetOutXMLFile("name",2,"data")        4
return render_template('mainpanel.html',                             5
                        NameTempSensor1=NameTempSensor1,             6
                        NameTempSensor2=NameTempSensor2)             7
```

Pro uložení nastavených hodnot do xml souboru je na html stránce *settings.html* vytvořen formulář. Jeho vyplnění a odeslání poté vede k zápisu nastavených hodnot do xml souborů, zobrazeno ve výpise 3.18. Ve formuláři se nacházejí vstupní prvky typu *textfield*, *checkbox* a *select*. Textové pole slouží k zapsání názvu popřípadě hodnoty. Checkbox nám vrací hodnoty povoleno/zakázáno a select vybíráme z přednastavený hodnot. Každý takový to vstup má název a id pro další manipulaci

s daty. Po nastavení všech potřebných hodnot se formulář potvrdí tlačítkem *Nastav* a dojde k odeslání požadavku.

Výpis 3.18: Formulář pro odeslání dat

```
<form  name="form1" method = "POST" action="/send" >
<input type="text" name="NameTempSensor1"
value="{{NameTempSensor1}}">
<input type="checkbox" id="CheckBoxArduinoInput1"
name="PermitInputArdudino1">
<input class="btn␣btn-primary" type="submit" value="Nastav">
</form>
```

Takto odeslaná data jsou zpracována. Pomocí frameworku je zachycen požadavek pomocí kontroly *reguest.for.get*, celý postup je zobrazen ve výpise 3.19. Pomocí podmínky je sledováno zda byl vyvolán požadavek typu POST. Pokud ano pokračuje se získání jednotlivých parametrů. Získané parametry jsou potom ukládány do xml souboru.

SetOutXMLFile()

Tato vytvořená funkce je obdobou funkce *GetOutXMLFile()* s tím rozdílem, že po naparsování dat dojde přepsání hodnoty určitého elementu. Poté následuje uložení celých dat do souboru, ze kterého byly čteny.

Výpis 3.19: Odeslání dat

```
@app.route('/send', methods=['POST', 'GET'])
def send():
 if request.method == 'POST':
  NameOutput1 = request.form.get('NameOutput1')
  NameOutput2 = request.form.get('NameOutput2')
  pi_control.SetOutXMLFile("name",1,NameOutput1,"data_output")
  pi_control.SetOutXMLFile("name",2,NameOutput2,"data_output")
 return redirect(url_for('settings'))
```

Výše je popsáno předávání statických parametrů mezi jednotlivými webovými stránkami. Na hlavním panelu je však potřeba zobrazovat data dynamicky bez obnovení stránky. To je zajištěno pomoci funkce vytvořené v JavaSkriptu, uvedeno ve výpise 3.20. Vytvořená funkce využívá *XMLhttpRequest* objekt pro vyžádání dat ze serveru bez opětovného načtení stránky. Starší prohlížeče nedisponují tímto objektem, proto je tato podmínka vyhodnocována a pro starší prohlížeče je vytvořen *ActiveXobject*. Vlastnost *onreadystatechange* určuje funkci, která má být provedena při každé změně. Tedy otevření souboru a načtení obsahu. Poté následuje získání konkrétního prvku.

Výpis 3.20: Načítání dynamického obsahu pomocí JavaScriptu

```
function ReadTemperatureFromXML() {                                    1
  var xmlDoc;                                                          2
  var xmlhttp;                                                         3
  if (window.XMLHttpRequest) {          //Mozila,Safari,IE+7           4
    xmlhttp = new XMLHttpRequest(); }  //vytvoření požadavku           5
  else {xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");              6
  // IE 6 and older}                                                   7
  if (!xmlhttp) {alert(Cannot create an XMLHTTP instance);             8
                 return false;}                                        9
  xmlhttp.onreadystatechange=datasend();{                              10
  //akce po odpovědi serveru                                           11
  function datasend(){                                                 12
  xmlhttp.open("GET", "data.xml", false); //otevření souboru           13
  xmlhttp.send();                          //zaslaní obsahu            14
            }                                                          15
```

3.6 Záznam dat

Pro pozdější náhled naměřených dat, je využito funkce databáze. Na Raspberry je nainstalován MySQL server MariaDB, příkazy zobrazeny ve výpise 3.21. Pro lepší náhled a práci s databází je dále nainstalován *Phpmyadmin*, kde je možné pomocí webového aplikace spravovat databázi. Pro každý vstup byla vytvořena tabulka. Základními parametry těchto tabulek tvoří id, hodnota veličiny a čas přijatých dat. Časový záznam je generován automaticky. Jedna z vytvořených tabulek je zobrazena na obr. 3.9.

Výpis 3.21: Instalace MariaDB a phpmyadmin

```
sudo apt-get install mariadb-server //instalace mariadb-server   1
apt-get install phpmyadmin          //instalace phpmyAdmin       2
```

	#	Název	Typ	Porovnávání	Vlastnosti	Nulový	Výchozí	Komentáře	Další
☐	1	id	int(1)			Ne	*Žádná*		
☐	2	Temperature	int(4)			Ne	*Žádná*		
☐	3	insDT	datetime		on update CURRENT_TIMESTAMP	Ne	CURRENT_TIMESTAMP		ON UPDATE CURRENT_TIMESTAMP

Obr. 3.9: Základní struktura databázových tabulek

Pro práci s databázi jsou vytvořeny základní funkce:

GetDataDatabaseInt()

Základním parametrem této funkce je ComString, kde je napsán příkaz pro komunikaci s vytvořenou databází. Vytvořená funkce se spojí s lokální databází a zapíše zadaný dotaz. Výsledkem je navrácení hodnot v případě úspěšného spojení a existenci dotazovaných dat a korektní ukončení spojení s databází. Zdrojový kód je zobrazen ve výpise 3.22.

Výpis 3.22: Funkce získání dat z databáze

```
def GetDataDatabaseInt(self, ComString):                          1
  db = MySQLdb.connect("localhost","root",                         2
  "raspberry","develop" )                                          3
  //metoda s argumenty sql databaze                                4
  cur = db.cursor()                                                5
  try:                                                             6
    cur.execute(ComString) //dotaz                                 7
    data = [item[0] for item in cur.fetchall()]                    8
    //získání jednotlivých dat                                     9
  except:              //pokud nejsou data                         10
    data = 0           //nulová hodnota                            11
  db.close()           //ukončení spojení                          12
  dataout=[]           //inicializace výstupu                      13
  for s in data:       //pro všechny data                          14
      s=int(s)         //přetypování                               15
     dataout.append(s)//uložení dat do seznamu                     16
  return dataout       //návrat dat                                17
```

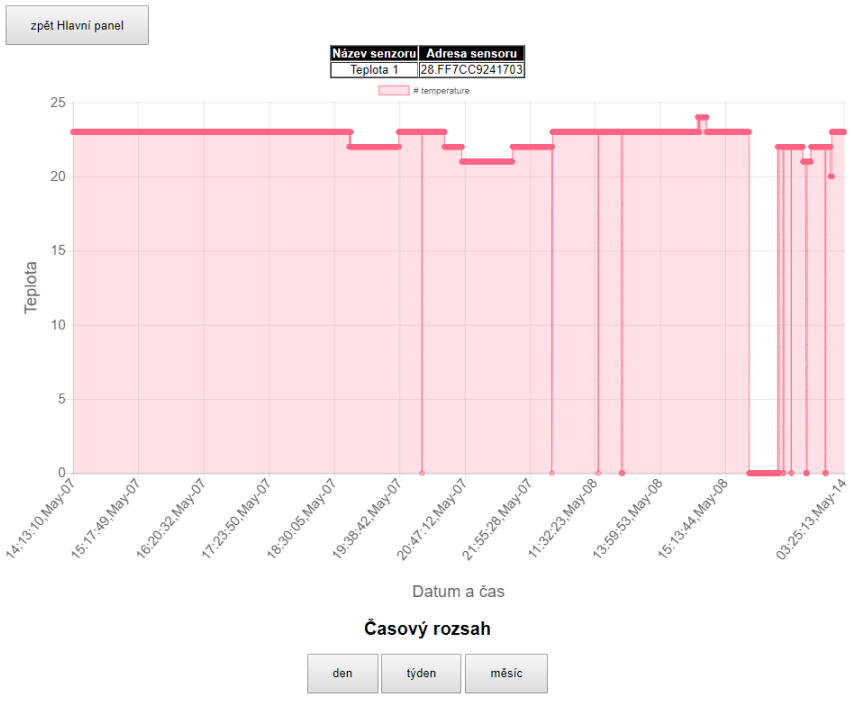
InsertDataDatabase()

Tato funkce slouží k zápisu dat do databáze. Funkce je obdobná funkci pro získání hodnot z databáze. Zde je zapsán pouze konkrétní příkaz pro zápis dat do příslušné vytvořené tabulky. Jak je uvedeno ve výpise 3.23.

Výpis 3.23: Funkce vložení dat do databáze

```
def InsertDataDatabase(self, ComString, value1, value2):
  db = MySQLdb.connect("localhost", "root",
  "raspberry", "develop" )
  cur = db.cursor()
  value1 = str(value1)      //přetypování
  value2 = str(value2)      //přetypování
  try:                      //zkus vložit data
    cur.execute("INSERT␣INTO␣"+ComString+"VALUES(\%s,\%s)",
    (value1, value2))
    db.commit()             //uložení změn v databázi
  except:                   //pokud nastane vyjímka
    db.rollback()           //zrušení transakce
  db.close()
```

Pro zobrazení zaznamenaných dat je využito objektu na hlavním panelu, kde jsou zobrazovány aktuální data ze senzorů. Stiskem na příslušný blok je načtena html stránka s naměřenými daty pro aktuální den. Dále se zde nacházejí tlačítka pro změnu vypisovaných dat a to v intervalu den, týden a měsíc. Tyto data jsou vynesena do grafu zobrazeno na obr. 3.10. Pro tvorbu grafu je využito JavaScriptové knihovny *Charts.js*, která umožňuje tvorbu animovaných a interaktivních grafů na webový stránkách.

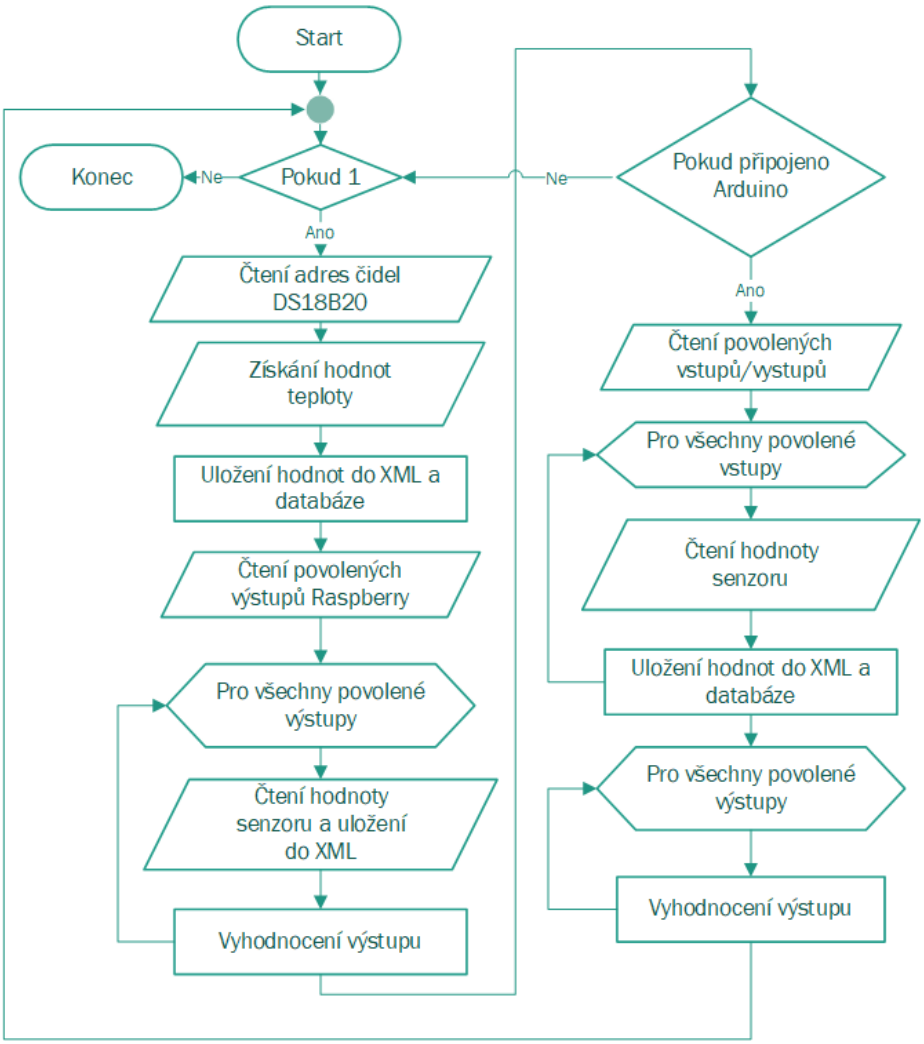


Obr. 3.10: Vizualizace naměřených hodnot

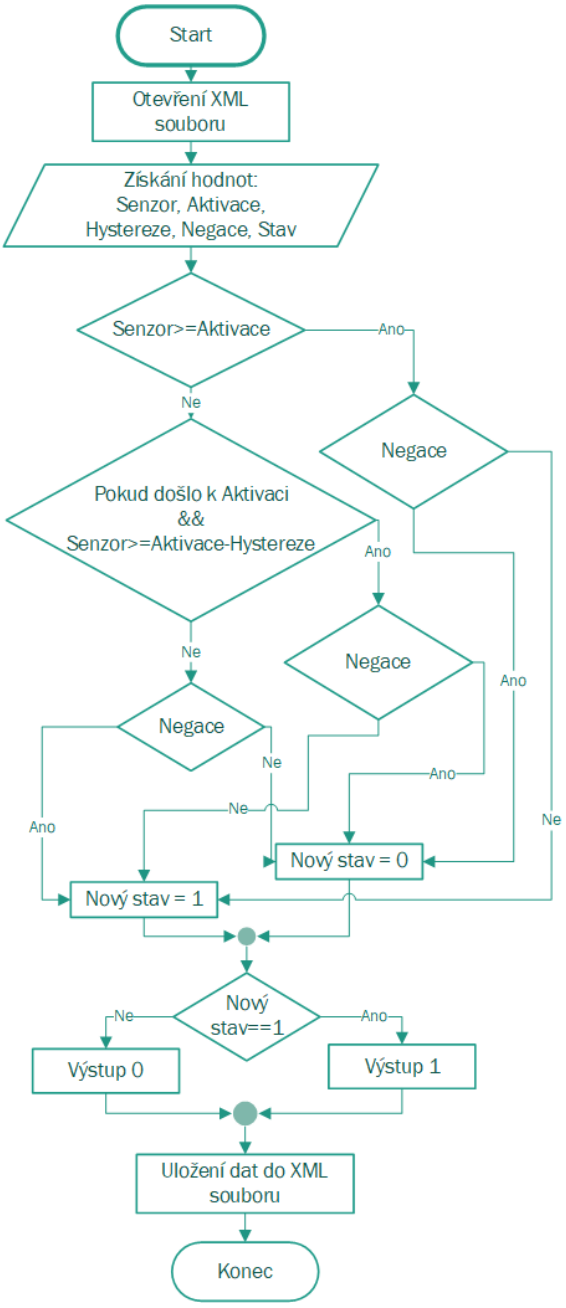
3.7 Hlavní proces

Na pozadí webového rozhraní běží python skript, který pracuje s vytvořenými obslužnými funkcemi. S jeho pomocí jsou do XML souborů ukládány aktuální informace ze senzorů a vyhodnocovány podmínky pro aktivaci výstupů. Vývojový diagram je zobrazen na obr. 3.11.

Nejprve jsou získány adresy připojených čidel, nasleduje její vyčtení a uložení do souboru. Dále je kontrolováno zda je připojeno Arduino na získání adresy, pokud je získána adresa, provede se zaslání požadavku na konkrétní hodnoty z aktivovaných senzorů. Poté následuje kontrola zda jsou aktivní výstupy. Pokud jsou aktivní, přečte se hodnota adresy čidla, které je danému výstupu přidělen a získá se poslední přijatá hodnota z XML souboru a uloží se do daného souboru k výstupu. Na to algoritmus aktivace výstupu zkontroluje, zda má dojít ke změně stavu zobrazeno ve vývojovém diagramu na obr. 3.12. To stejné probíhá i pro připojené Arduino a jeho výstupy.



Obr. 3.11: Vývojový diagram řídícího procesu



Obr. 3.12: Vývojový diagram vyhodnocení výstupu

Vytvořené skripty jsou spouštěny po nabootování Raspberry. Spouštěcí funkce je uložena do složky s cestou */etc/rc.local*, kde jsou na konec souboru zapsány spouštěcí příkazy pro vytvořené skripty zobrazeno ve výpise 3.24 .

Výpis 3.24: Nastavení spouštění skriptů po nabootování Raspberry

```
sudo python2.7 Deamon.py &                                    1
sudo python2.7 Kotelna.py &                                   2
Exit 0                                                        3
```

4 ZÁVĚR

Cílem diplomové práce bylo sestavit elektronický systém pro Monitorování kotelen na tuhá paliva.

V první kapitole jsou popsány veličiny, které se obvykle v kotelnách na tuhá paliva monitorují. Dále jsou popsány principy senzorů, které se nejčastěji pro měření daných veličin používají. Jsou zde uvedeny současně na trhu dostupná zařízení, které se problémem monitoringu kotelen zabývají.

Ve druhé kapitole jsou nejprve vzneseny požadavky, které by měl vytvořený systém splňovat. Podle požadavků je potom navržená vhodná koncepce zařízení. Jsou vybrány vhodné hlavní řídící a senzorické prvky. Je navrženo elektronické zapojení senzorů a jsou uvedeny základní výpočty pro vybrané elektronické součástky s ohledem na správnou funkčnost zařízení. Po navržení základních obvodů je propočítána proudová spotřeba navrženého zařízení a je sestaven vhodný napájecí obvod, který zajistí dodání potřebné energie pro zařízení.

Třetí kapitola se zabývá návrhem obslužného softwaru. Skripty a programy pro jednotku Raspberry jsou napsané v programovacím jazyce python, program pro ovládání Arduina je napsán pomocí knihovny Wiring. Grafické rozhraní pro ovládání systému je vytvořeno v jazyce HTML s využitím některých funkcí JavaScriptu. V grafickém rozhraní je možné sledovat aktuální naměřené hodnoty senzorů a je možné nastavovat reakce připojených akčních členů na naměřené hodnoty.

Vytvořené zařízení bylo otestováno. Pomocí grafického rozhraní je možné nastavovat jednoduché ovládací smyčky hodnota senzoru - výstup. Naměřená data ze senzorů jsou ukládány do databáze a je možné graficky zobrazovat denní, týdenní a měsíční záznam.

LITERATURA

[1] BURDA, Karel a Ivo STRAŠIL. *Zabezpečovací systémy.* Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií 2012. ISBN 978-80-214-4441-6.

[2] Synex Controls Inc. *The intelligent solution for control and sequencing of multiple boiler systems.* [online]. [cit. 13. Prosince 2017]. Dostupné z URL: <http://www.synexcontrols.com/insite.php>.

[3] Siemens. *Poruchová signalizace kotelník v1.0.* [online]. [cit. 13. Prosince 2017]. Dostupné z URL: <https://www.bola.cz/admin/files/e_product_files/2/2056/Navod_poruchovka_V01_2017_1_16.pdf>.

[4] MAIXNER, Ladislav. *Mechatronika*: učebnice. Brno: Computer Press, 2006. Učebnice (Computer Press). ISBN 80-251-1299-3.

[5] KADLEC Karel a Miloš KMÍNEK. *Měřicí a řídicí technika* :elektronický učební text VŠCHT Praha, 2001.

[6] MATUŠKA, Tomáš. *Experimentální metody v technice prostření* Praha: Česká technika - nakladatelství ČVUT, 2005. ISBN 80-01-03291-4.

[7] HŮNOVÁ, Iva a Svatava JANOUŠKOVÁ. *Úvod do problematiky znečištění venkovního prostředí.* Praha: Karolinum, 2004. ISBN 80-246-0796-4.

[8] PROTRONIX s.r.o. *Čidla kvality vzduchu.* [online]. [cit. 13. Prosince 2017]. Dostupné z URL: <https://www.careforair.eu/na-jakych-principech-/funguji-cidla-kvality-vzduchu/>.

[9] *Ndir carbon dioxide sensing technology.* Application note TSI-037. [online]. [cit. 12. Prosince 2017]. Dostupné z URL: <http://www.tsi.com/uploadedFiles/_Site_Root/Products/Literature/Application_Notes/TSI-037.pdf>.

[10] Combustion Controls Center Europe. *Flame current measurement.* Application note AN 3. [online]. [cit. 2. Října 2017]. Dostupné z URL: <http://www.honeywell.cz/combush/OdborneClanky/AN3.pdf>.

[11] SCOTT INSTRUMENT. *A Guide to Selecting the Right Flame Detector For Your Application.* [online]. 2006 - [cit. 14. prosince 2006]. Dostupné z URL: <http://www.microtech.co.th/pdf/flame-detector.pdf>.

[12] ZEZULKA, František. *Prostředky průmyslové automatizace.* Brno: VUTIUM, 2004. 176 s. ISBN: 80-214-2610-1

[13] *Boiler Remote Monitoring and Alarm Notifications*[online].[cit. 10. května 2018]. Dostupné z URL:
<https://remforce.com/>.

[14] UPTON Eben a Gareth HALFACREE. *Raspberry Pi User Guide.*[online].[cit. 2. Října 2017]. Dostupné z URL:
<http://www.cs.unca.edu/~bruce/Fall14/360/RPiUsersGuide.pdf>.

[15] Maxim Integrated Products, Inc. *Single-Channel 1-Wire Master*. [online]. [cit. 1. Prosince 2017]. Dostupné z URL:
<https://datasheets.maximintegrated.com/en/ds/DS2482-100.pdf>.

[16] Texas Instruments Inc. *High-Voltage, High-Current Darlington Transistor Arrays*. [online].[cit. 1. Prosince 2017]. Dostupné z URL:
<https://datasheets.maximintegrated.com/en/ds/DS2482-100.pdf>.

[17] AOSONG.*Temperature and humidity module.*[online].[cit. 15. Prosince 2017]. Dostupné z URL:
<https://akizukidenshi.com/download/ds/aosong/DHT11.pdf>.

[18] Maxim Integrated Products, Inc. *Cold-Junction-Compensated K-Thermocoupleto-Digital Converter (0°C to +1024°C)* Application note AN 3. [online]. [cit. 3. března 2018]. Dostupné z URL:
<https://datasheets.maximintegrated.com/en/ds/MAX6675.pdf>.

[19] Zhengzhou Winsen Electronics Technology Co. *Toxic Gas Sensor*. [online]. [cit. 12. Prosince 2017]. Dostupné z URL:
<https://cdn.sparkfun.com/datasheets/Sensors/Biometric/MQ-7%20Ver1.3%20-%20Manual.pdf>.

[20] Olimex. *MQ-135 GAS SENSOR*. [online]. [cit. 12. Prosince 2017]. Dostupné z URL:
<https://www.olimex.com/Products/Components/Sensors/SNS-MQ135/resources/SNS-MQ135.pdf>.

[21] Cadex Electronics Inc.*BU-808: How to Prolong Lithium-based Batteries.* Application note AN 3. [online]. [cit. 15. Října 2017]. Dostupné z URL:
<http://batteryuniversity.com/learn/article/how_to_prolong_lithium_based_batteries>.

[22] Texas Instruments Inc. *TPS6123x High Efficiency Synchronous Step Up Converters with 5-A Switches (Rev. C)*. [online].[cit. 1. Prosince 2017]. Dostupné z URL:
<http://www.ti.com/product/TPS61230/datasheet/abstract#SLVSAQ22997>.

[23] Microchip Technology Inc. *Stand-Alone Linear Li-Ion / Li-Polymer Charge Management Controller* . [online]. [cit. 1. Prosince 2017] Dostupné z URL:
<http://www.mouser.com/ds/2/268/22005a-76648.pdf>.

[24] Microchip Technology Inc. *Designing A Li-Ion Battery Charger and Load Sharing System With Microchip's Stand-Alone Li-Ion Battery Charge Management Controller* . [online]. [cit. 1. Prosince 2017] Dostupné z URL:
<http://ww1.microchip.com/downloads/en/AppNotes/01149c.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

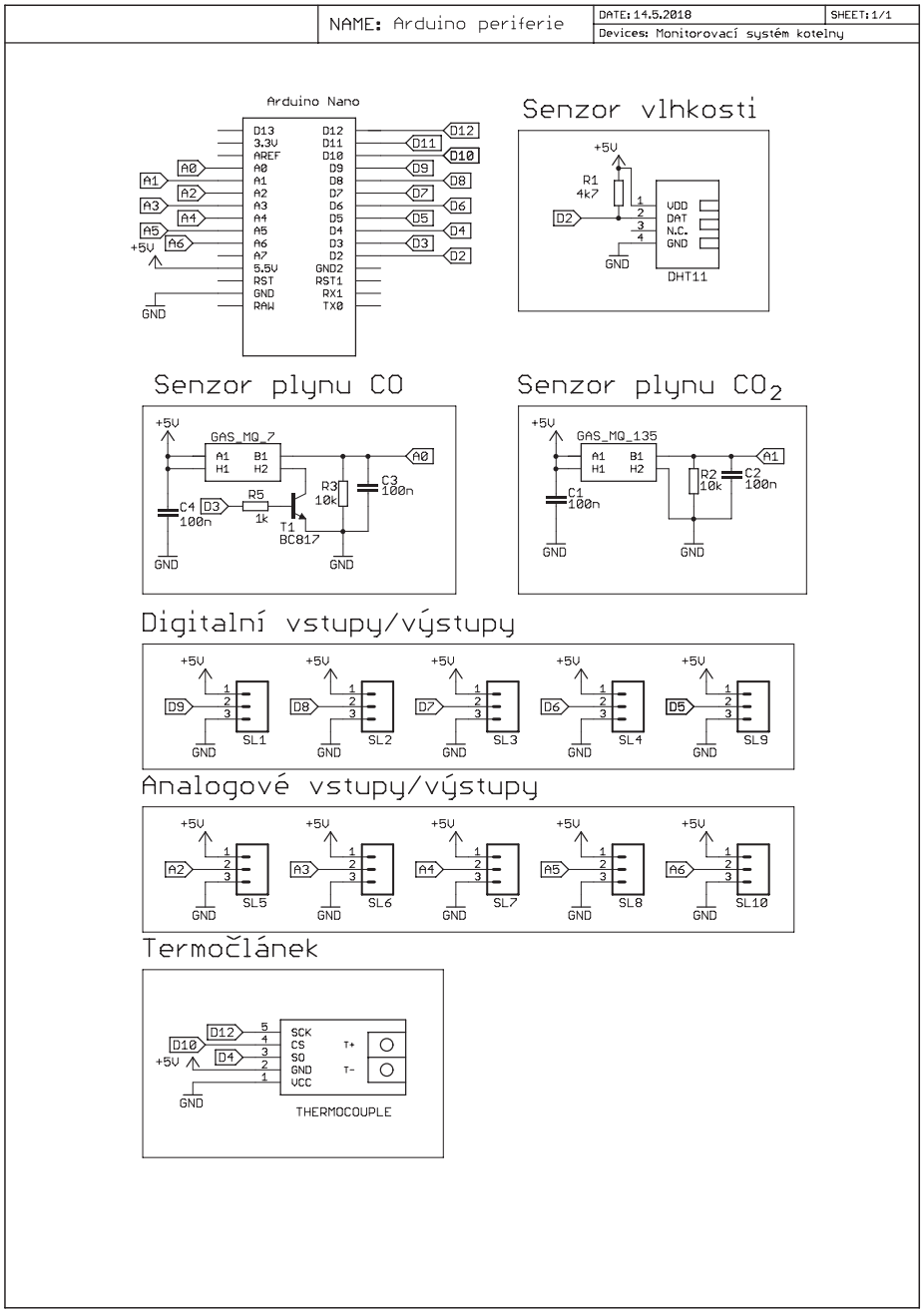
Symbol	**Anglický význam**	**Český význam**
A	(Amper)	Ampér jednotka proudu
ARM	(Acorn Risc Machine)	Architektury procesorů
CPU	(Central processing unit)	Procesor
C	(Carbon)	Uhlík
CO	(Carbon monoxide)	Oxid uhelnatý
CO_2	(Carbon dioxide)	Oxid uhličitý
EPS	(-)	Elektronická požární signalizace
GPIO	(General-purpose input/output)	Vstupní/výstupní porty
HMI	(Human–Machine Interface)	Rozhraní člověk stroj
HVAC	(Heating, ventilation,air-con)	Vytápění, ventilace, klimatizace
h_{FE}	(-)	Zesílení tranzistoru
I	(-)	Symbol elektrického proudu
I_{C}	(-)	Kolektorový proud
I_{LED}	(-)	Proud světelnou diodou
IoT	(Internet of things)	Internet věcí
IP	(Internet protocol)	Internetový protokol
IR	(Infrared)	Infračervené záření
I_{REG}	(-)	Nabíjecí proud
I_{VYST}	(-)	Výstupní proud
I_{1WIRE}	(-)	Proud sběrnice 1-Wire
$\mathrm{I^2C}$	(Inter-Integrated Circuit)	Sběrnice
LED	(Light Emitting Diode)	Světelná dioda
Li-Ion	(Lithium-ion battery)	Lithium-iontová baterie
Li-Pol	(Lithium-polymer battery)	Lithium-polymerová baterie
NDIR	(-)	Měřící článek
O	(Oxid)	Kyslík
PLC	(Programmable logic controller)	Programovatelný logický automat
PWM	(Pulse Width Modulation)	Pulzně šířková modulace
ppm	(Parts per million)	Částice na jeden milion
R	(-)	Symbol elektrického odporu
R_{VIN}	(-)	Odpor vinutí cívky relé
SCADA	(Supervisory Control and Data-Acquisition)	Supervizní řízení a sběr dat
SPI	(Serial Peripheral Interface)	Seriové periferní rozhraní
U	(-)	Symbol elektrického napětí
U_{FB}	(-)	Výstupní napětí referenční TPS61230

U_{GPIO}	(-)	Napětí výstupu Raspberry
$U_{\mathrm{IN\_ON}}$	(-)	Spouštěcí napětí
$U_{\mathrm{IN\_OFF}}$	(-)	Vypínácí napětí
U_{LED}	(-)	Úbytek napětí na LED
U_{OPTO}	(-)	Úbytek napětí na optočlenu
U_{RAS5V}	(-)	Napětí Raspberry 5 V
U_{RAS33V}	(-)	Napětí Raspberry 3,3 V
U_{UBDS24}	(-)	Úbytek napětí konvertoru DS2482
$U_{\mathrm{TH\_ON}}$	(-)	Referenční napětí TPS61230
$U_{\mathrm{TH\_OFF}}$	(-)	Referenční napětí TPS61230
USB	(Universal Serial Bus)	Univerzální sériová sběrnice
UV	(Ultraviolet)	Ultrafialové záření
U_{VSAT}	(-)	Saturační napětí tranzistoru
U_{VST}	(-)	Vstupní napětí
U_{VYST}	(-)	Výstupní napětí
V	(Volt)	Jednotka napětí
W	(Watt)	Jednotka výkonu
Wifi	(-)	Standart pro přenost dat
Ω	(Ohm)	Jednotka odporu
1-Wire	(-)	Datová sběrnice

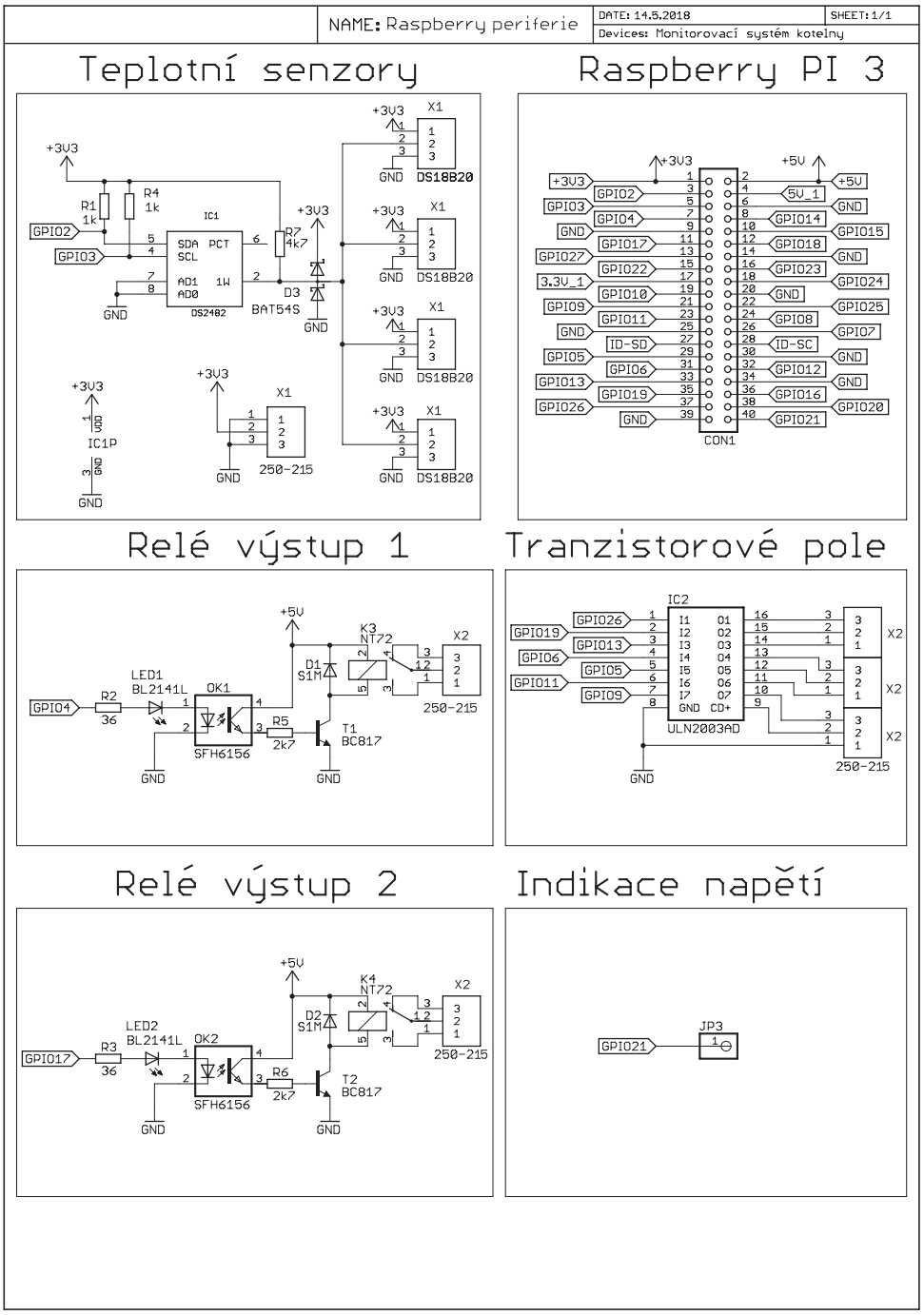
SEZNAM PŘÍLOH

A SCHÉMATA

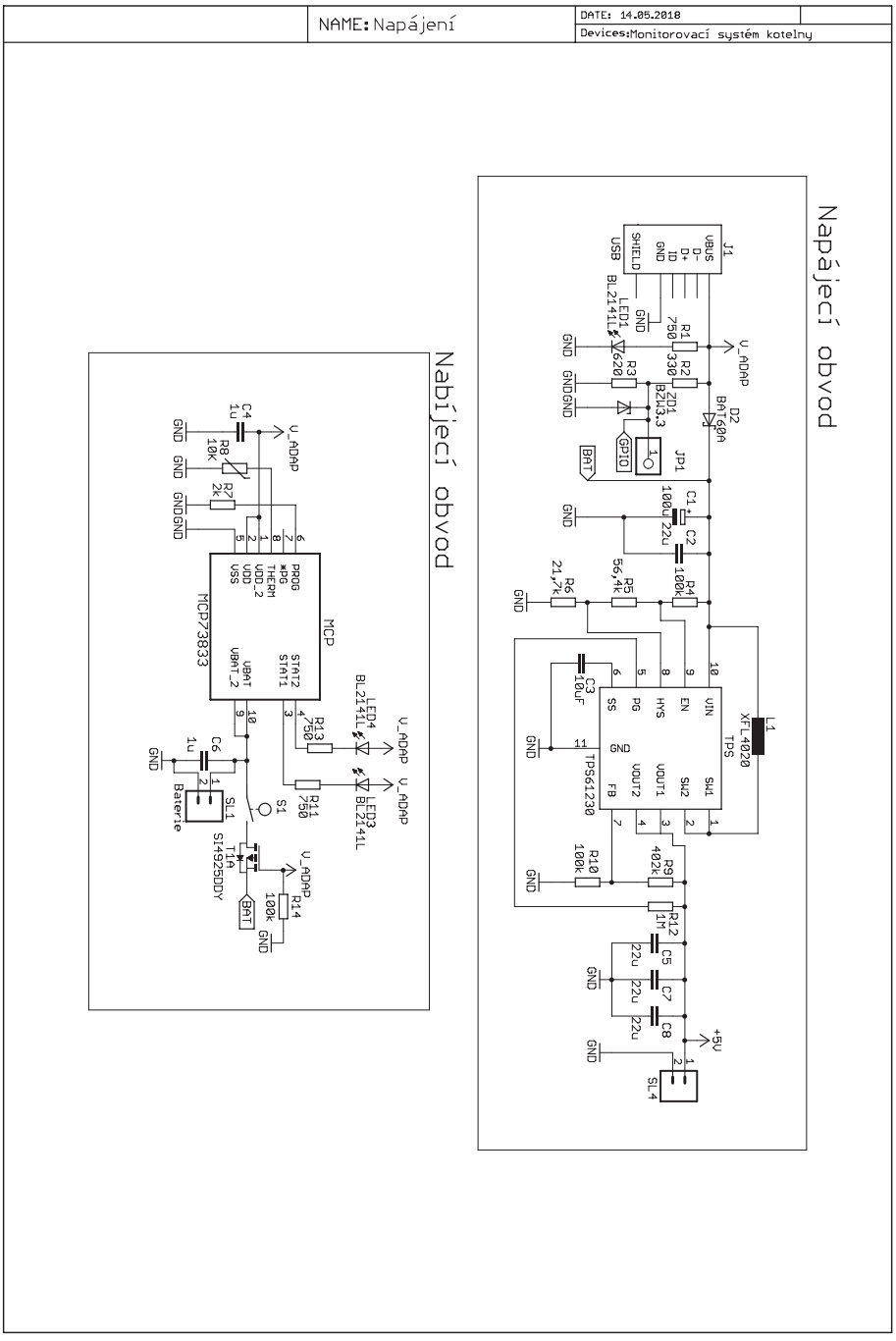
A.1 Arduino periferie



A.2 Raspberry periferie



A.3 Napájení


NAME: Napájení
DATE: 14.05.2018
Devices: Monitorovací systém kotelny
Napájecí obvod
Nabíjecí obvod

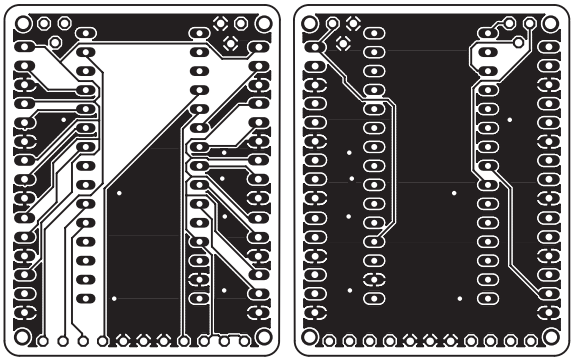
B SEZNAM SOUČÁSTEK

Arduino periferie				
Součást	**Hodnota**	**Pouzdro**	**Kód Gme/Far**	**Cena**
SL1-SL10	ARK500	M03	821-557	190,00 Kč
Arduino Nano	Arduino Nano	-	772-238	160,00 Kč
R5	1k	R1206	900-033	2,00 Kč
R2, R3	10k	R1206	901-176	2,00 Kč
R1	4,7k	R1206	900-120	1,00 Kč
T1, T2	BC817-40	SOT23	912-148	4,00 Kč
U1	DHT11	-	772-137	79,00 Kč
GS1	MQ-135	-	-	30,00 Kč
GS2	MQ-7	-	-	30,00 Kč
C1-C4	100n	-	120-060	6 Kč
Raspberry periferie				
Součást	**Hodnota**	**Pouzdro**	**Kód Gme/Far**	**Cena**
CON1		2x20pin	832-009	24,00 Kč
X1,X2	Wago 250-215	3,50 mm	820-346	128,00 Kč
LED1, LED2	BL-B2141-L	3 mm	511-200	6,00 Kč
JP1		1pin	832-021	1,00 Kč
R1, R4	1k	R1206	900-033	2,00 Kč
R2, R3	36	R1206	900-118	2,00 Kč
R7	4k7	R1206	900-120	1,00 Kč
R5, R6	2k7	R1206	900-182	2,00 Kč
T1, T2	BC817-25	SOT23	912-148	2,00 Kč
IC1	DS2482-100S	SO8	959-491	51,00 Kč
K4,K3	NT72-2	-	634-177	58,00 Kč
D1, D2	S1M	S1M	917-036	3,00 Kč
OK1, OK2	SFH6156	DIP4 SMD	961-059	12,00 Kč
IC2	ULN2003AD	SO16	930-005	10,00 Kč
D3	BAT54S	SOT-23	920-103	1,50 Kč

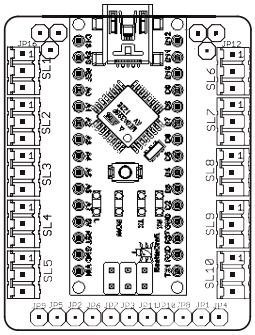
Napájení				
Součást	**Hodnota**	**Pouzdro**	**Kód Gme/Far**	**Cena**
C1	100u	Radiální	123-212	2,00 Kč
C2,C5,C7,C8	22u	C1206	906-145	4,00 Kč
C3	10u	C1206	905-029	1,00 Kč
C4,C6	1u	C1206	905-136	4,00 Kč
D2	BAT60A	SOD323	920-116	5,00 Kč
LED1,3,4	BL-B2141-L	3 mm	511-200	9,00 Kč
JP1		PINHD-1X1	832-021	1,00 Kč
L1	XFL4020	MS42	2289214	71,00 Kč
R1	750	R1206	900-450	1,00 Kč
R2	330	R1206	900-140	1,00 Kč
R3	620	R1206	900-087	1,00 Kč
R4,R10,R14	100k	R1206	900-003	2,00 Kč
R5	56,4k	R1206	-	1,00 Kč
R6	21,7k	R1206	-	1,00 Kč
R7	2k	R1206	900-070	10,00 Kč
R8	NTC-10K	NTC 640-10K	118-042	12,00 Kč
R9	402k	R1206	-	1,00 Kč
R11,R13	750	R1206	900-450	2,00 Kč
R12	1M	R1206	901-036	1,00 Kč
SL1,SL4, S1	AKZ962	M02	821-357	39,00 Kč
T1	Si492DDY	SO8	915-035	19,00 Kč
TPS	TPS61230	VSSOP-10	2782634	65,00 Kč
MCP	MCP73833T	DFN-10	2709765	21,00 Kč
J1	USB	USB-B	832-239	37,00 Kč
ZD1	BZW55C3.3	SOD80C	919-045	2,00 Kč

C PODKLADY PRO VÝROBU DPS

C.1 Arduino periferie

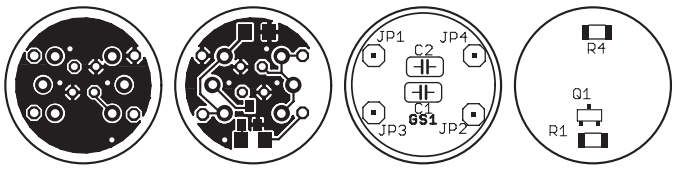


Obr. C.1: Deska plošného spoje pro Arduino vrchní, spodní strana



Obr. C.2: Plán osazení Arduino

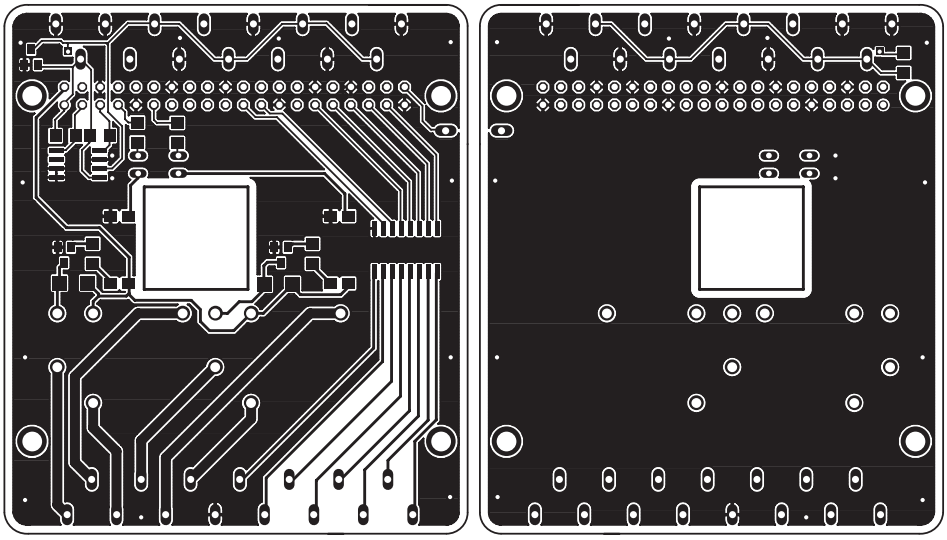
Z důvodu ušetření místa je pro plynové senzory zvlášť vytvořen plošný spoj, kde je umístěna elektronika. Plošný spoj je k Arduinu připojen pomocí vodičů.



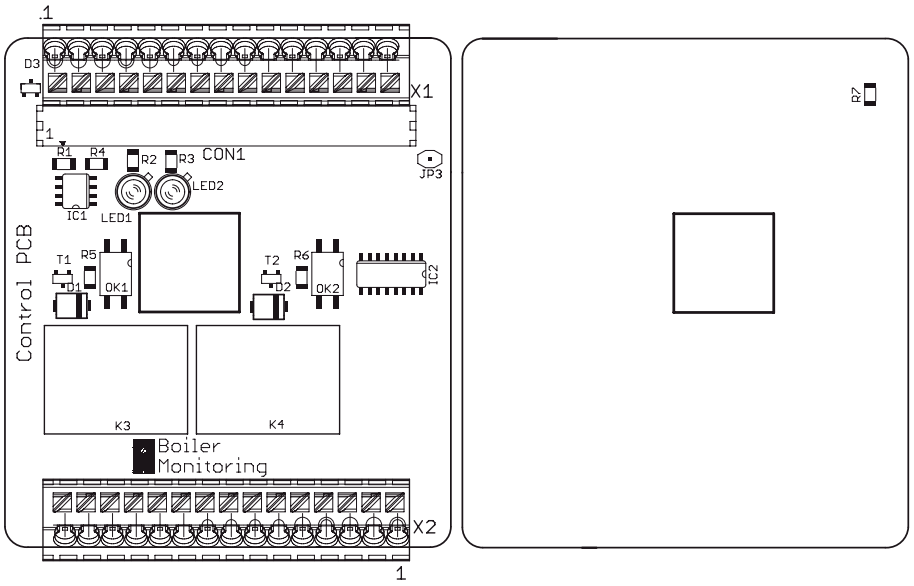
Obr. C.3: Deska plošného spoje pro senzory plynu vrchní, spodní strana a osazení vrchní a spodní strany

C.2 Raspberry periferie

Uprostřed desky je otvor pro zajištění průchodu vzduchu ke chladiči procesoru u Raspberry.

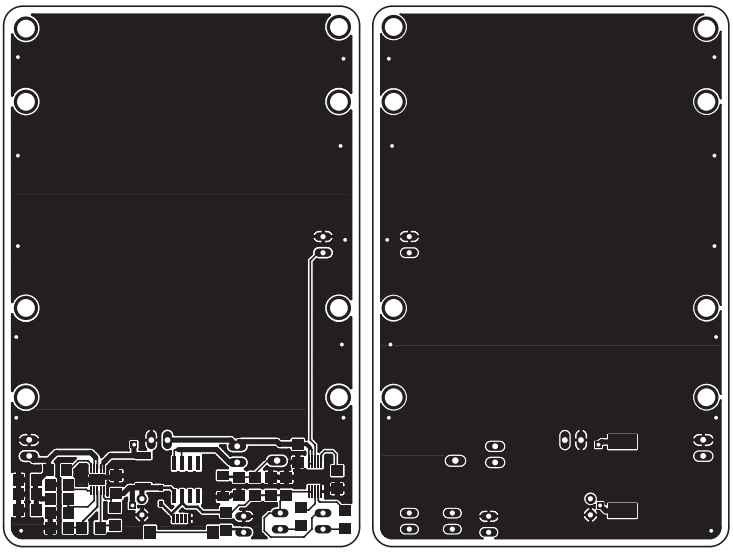


Obr. C.4: Navžená deska plošného spoje Raspberry vrchní a spodní strana

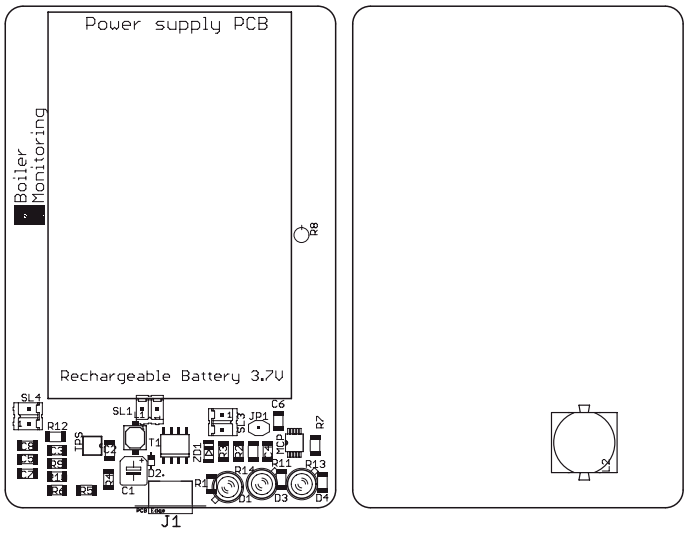


Obr. C.5: Osazení desky pro Raspberry

C.3 Napájení



Obr. C.6: Napájecí deska plošného spoje vrchni a spodní strana



Obr. C.7: Osazení desky pro napájení.

D OBSAH PŘILOŽENÉHO SOUBORU

V přiloženém souboru se nachází vytvořený program pro Raspberry a Arduino. Dále jsou přiloženy soubory se schématy a návrhy desek plošných spojů, vytvořené v návrhovém programu eagle.

- `/` ..... kořenový adresář
  - `DP_Schema` ..... eagle soubory
  - `Arduino_code` ..... program pro Arduino
  - `Raspberry_code` ..... program pro Raspberry